National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

NL-339 (r. 88/04) c

Canada

University of Alberta

A GATE-ARRAY DESIGN APPROACH FOR AN ANALOG TO DIGITAL INTERFACE FOR DATA ACQUISITIONS AND TRANSFERS

by

Makarand Paranjape

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree
of Master of Science

Department of Electrical Engineering

Edmonton, Alberta
Fall, 1990

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-65070-2

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Makarand Paranjape

TITLE OF THESIS: A Gate-Array Design Approach for an Analog to Digital Interface for Data Acquisitions and

DEGREE FOR WHICH THIS THESIS WAS PRESENTED: Master of Science

YEAR THIS DEGREE GRANTED: 1990

Permission is hereby granted to The University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(Signed) M. Paranjape

Permanent Address:
10356-146th Street
Edmonton, Alberta
Canada

Dated: Sept. 6, 1990

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled **A Gate-Array Design Approach for an Analog to Digital Interface for Data Acquisitions and Transfers** submitted by **Makarand Paranjape** in partial fulfillment of the requirements for the degree of **Master of Science.**

Keith G. Stromsmoe

Dr. K. Stromsmoe - Supervisor

Dr. J. Leonard

Dr. P. Smy

Date

6 Sept 1990

Abstract

The research outlined in this thesis involves the design of a logic gate-array I.C. which can be used in certain data handling applications. The design allows for fast digital data acquisitions and the transfer of captured data at significantly slower rates. With these features in mind, this device is well-suited for use as a principle module in digital storage oscilloscope (DSO) systems. Furthermore, the A.D.I.D.A.T., an acronym for analog-to-digital interface for data acquisitions and transfers, makes use of a novel approach in its ability to capture pretrigger information. This is a highly desireable feature in DSO's, and can only be accomplished using digital techniques. Thus, with all control logic circuitry implemented on a single gate array, the ADIDAT-based digital storage scope can provide a compact, efficient, and cost-effective system.

Since this document only outlines the design of the ADIDAT module, one out of a series of comprehensive simulations of the system is included. This will be used as a means to analyze the resulting circuit performance.

Acknowledgements

I would like to thank my supervisor, Dr. Keith Stromsmoe, for his guidance and support throughout this research project. Also, I would like to thank Dr. Lj Ristic, Dr. P. Smy, and Dr. R. Lawson for their invaluable help with this thesis. In this endeavor, many people contributed helpful suggestions to problems and hurdles which had to be overcome. In this respect, I would like to thank Steve Slupsky (AMC-Applications Engineer), Randy White (AMC-Technologist), Norman Jantz (Sun System Manager), Mladen Rajhard, Stuart Olsen, Melinda Wilson, and Mike Anton. Most importantly, however, I would like to thank my parents, who were not only encouraging and supportive throughout this project, but also showed a great deal of understanding and patience during my university education.

Table of Contents

1. Introduction 1
  1.1. Background 4
    1.1.1. Concepts of an Analog Oscilloscope 4
    1.1.2. Concepts of the Digital Storage Oscilloscope 7
    1.1.3. Introduction to Sampling 10
    1.1.4. Flash Analog-to-Digital Converters 15
2. Designing the ADIDAT 20
  2.1. Previous Work 20
  2.2. Design Considerations 20
  2.3. Design Procedure 22
  2.4. System Overview 26
  2.5. The ADIDAT-Based DSO 27
  2.6. Digital Storage Oscilloscope Operation 27
    2.6.1 Capture Control 30
    2.6.2 Refresh 30
    2.6.3 Roll 31
    2.6.4 Pretrigger Options 32
    2.6.5 Storage Record Size 33
    2.6.6 Memory Block Selector 33
    2.6.7 Long Record Option 35
    2.6.8 Memory Block Loading 35
    2.6.9 Store, Clear, and Recall 36
    2.6.10 Timebase Selection 37
    2.6.11 Execution Control 37
    2.6.12 The Keypad 40
    2.6.13 Memory Elements 44
3. Principles of Waveform Storage and Display 49
  3.1 Storage With 0% Pretrigger 51
  3.2 Storage With 25% Pretrigger 52
    3.2.1 Triggering after pretrigger counter reaches terminal count 52
    3.2.2 Triggering before pretrigger counter reaches terminal count 52
  3.3 Storage With 50% Pretrigger 53
    3.3.1 Triggering after pretrigger counter reaches terminal count 53
    3.3.2 Triggering before pretrigger counter reaches terminal count 53
  3.4 Storage With 75% Pretrigger 54
    3.4.1 Triggering after pretrigger counter reaches terminal count 54
    3.4.2 Triggering before pretrigger counter reaches terminal count 54

3.5 Storage With 100% Pretrigger 55
3.5.1 Triggering after pretrigger counter reaches terminal count 55
3.5.2 Triggering before pretrigger counter reaches terminal count 55
3.6 Recalling With 25% Pretrigger 59
3.6.1 Triggering after pretrigger counter reaches terminal count 59
3.6.2 Triggering before pretrigger counter reaches terminal count 59
4. Major Circuit Blocks of the ADIDAT 61
4.1. Pretrigger Control Circuitry 61
4.1.1 0% Pretrigger 65
4.1.2 25% Pretrigger 72
4.1.3 50% and 75% Pretrigger 73
4.1.4 100% Pretrigger 73
4.1.5 Increased Record Lengths 74
4.1.6 Other Circuit Components 74
4.2. The Memory Counter Block 76
4.3. Memory Addressing and RAM Control Circuitry 80
4.4. Prescalers 87
4.5. Latches 88
4.5.1 Pre-Latch Module 89
4.5.2 Post-Latch Module 91
4.5.3 Time-Latch Module 91
4.6. Counters 92
4.7. Timebase Generation Circuitry 94
4.8. Control Circuitry for the External L.E.D.'s 98
5. Testing the ADIDAT 103
5.1. Design Implementation and Procedure 103
5.2. Simulation Techniques 110
5.3. Simulation Results 111
6. Conclusions 118
7. References 120
8. Appendicies 123

List of Figures

Figure 1.1: A Simplified Block Diagram of an Analog Oscilloscope ..... Page 6

Figure 1.2: An Example Showing Pretrigger Information ..... Page 8

Figure 1.3: Signal Measurement Using Varying Sampling Rates ..... Page 11

Figure 1.4: Sampling at the Nyquist Rate ..... Page 12

Figure 1.5: Sampling at Twice the Nyquist Rate ..... Page 13

Figure 1.6: Sampling at an Adequate Rate ..... Page 14

Figure 1.7: Block Diagram of the Flash ADC ..... Page 18

Figure 1.8: Pinout for Samsung's KSV3100A-8 ADC ..... Page 19

Figure 2.1: The Basic Y-Chart ..... Page 23

Figure 2.2: The Y-Chart Used in the Design Process ..... Page 25

Figure 2.3: A General Block Diagram for a Digital Storage Oscilloscope ..... Page 28

Figure 2.4: The DSO Using the ADIDAT Module ..... Page 29

Figure 2.5: Pretrigger Options ..... Page 39

Figure 2.6: The Keypad Format ..... Page 41

Figure 2.7: Timing Diagrams for the Static RAM ..... Page 47

Figure 2.8: Timing Diagrams for the Dual-Ported RAM ..... Page 48

Figure 4.1: The PRETRIGGER Module ..... Page 62

Figure 4.2: Exclusive-Or Circuitry for 0% Pretrigger ..... Page 67

Figure 4.3: Internal Reset Circuitry ..... Page 68

Figure 4.4: Pretrigger Enabling Circuitry ..... Page 69

Figure 4.5: Posttrigger Enabling Circuitry ..... Page 71

Figure 4.6: YD_NOT In The Pretrigger Module ..... Page 77

Figure 4.7: A General Block Diagram for the ADIDAT Memory Elements ..... Page 86

Figure 4.8: The PRE_LATCH Data Register ..... Page 90

Figure 4.9: I/O For the 32:1 Multiplexer ..... Page 95

Figure 4.10: Timebase Organization ..... Page 96

Figure 4.11: Layout of the LED's ..... Page 100
Figure 4.12: Block Diagram of the RAM_LED Module ..... Page 101
Figure 5.1: Steps Required to Complete a Logic Design ..... Page 104
Figure 5.2: Heirarchial Level Structure ..... Page 106
Figure 5.3: An Example Showing a Heirarchial Design ..... Page 108
Figure 5.4: The Heirarchy of the ADIDAT Module ..... Page 109
Figure 5.5: The Simulated Input Waveform ..... Page 114

GLOSSARY OF SYMBOLS AND TERMS

GENERAL TERMS:

ADIDAT	Analog To Digital Interface for Data Acquisitions and Transfers
A-D or A/D	analog-to-digital converter, or ADC
ASIC	Application Specific Integrated Circuit
CMOS	Complementary Metal Oxide Semiconductor
count	pretrigger counter is stored with this value when transferring data
CRT	Cathode Ray Tube
D-A or D/A	digital-to-analog converter, or DAC
DPR	Dual-Port RAM
DSO	Digital Storage Oscilloscope
DSP	Digital Signal Processor/Processing
ECL	Emitter-Coupled Logic
EPLD	Eraseable, Programmable Logic Device
f	frequency
hex	hexadecimal based number (base 16)
IC	Integrated Circuit
Kbyte	kilobyte, or 1024 bytes
LDS	Logic Design System (LSI Logic Corp.)
LED	light emitting diode
LSB	least significant bit
LSED	Logic Schematics EDitor (LSI Logic Corp.)
KHz	Kilohertz
MHz	Megahertz
MSB	most significant bit
MSI	medium-scale integration

mux	multiplexer
NET	nework level in a schematic using LSED
PC-board	printed-circuit board
posttrigger	the time after the arrival of a valid trigger
pretrigger	the time before the arrival of a valid trigger
R/W'	read/not write line
R/W	read/write line
RAM	Random Access Memory
RECORD LENGTH	the amount of memory used for a data record
REFRESH	a mode of storing input data
ROLL	a mode of storing input data
S-R flip-flop	a Set-Reset flip-flop
SCL	Simulation Control Language (LSI Logic Corp.)
SYM	symbol level in a schematic using LSED
TC	the terminal count value of a counter (ie TC=15 for a 4-bit counter)
VLSI	Very Large Scale Integration
XOR	exclusive-OR gate, or EXOR gate
XTAL	crystal oscillator

SIGNAL NAMES:

@	address lines to RAMs (Fig.3)
ACQCOMP	a signal generated indicating a complete acquisition sequence
ADDR	address lines generated for DPR (Fig.3)
BLK	4-bit variable assigning data to a memory block
BLK3	3 most significant bits of variable BLK
BLKLOAD	variable used to hold the BLK value
BLSB	least significant bit of variable BLK
CAP_DIS	an input signal which indicates a data capture or transfer sequence
CD	clear direct signal for latches
CLEAR	active-high signal used to clear variables in latches
CLK	input clock signal for the counters
CLR	clear signal for counters
CNTR_SEL	an input signal into "SELECTOR" which selects post/pre-trigger counter
CO	the carry-out signal generated by a 11-bit counter
D	data lines (Fig.3)
DATAIN	8-bit input lines into ADIDAT for the digitized signal
EN	enable signal for counters
EN_DEC	signal to enable the 4-bit shift register in "SELECTOR"
EN_MSB	active-high signal which enables the 11th bit of the counter
EXTCLK	an external, user-definable clock input to "CLKS"
GN	active-low enable signal for latches
LATCHED	signal used for latching data into "POST_LATCH"
LD	load signal for latches
LOAD	load signal for counters
MEM_SEL	input signal to select 1 Kbyte or 2 Kbyte record storage
PRE100	input signals used to set the pretrigger value

PRE50	input signals used to set the pretrigger value
PRE25	input signals used to set the pretrigger value
R_RESET	an internal-reset signal for the "PRETRIGGER" module
RECALL	signal used to load pretrigger terminal count into counters
SHORT_CO	the carry-out signal generated by a 10-bit counter
STORE	active-high signal used to hold variables in latches
TC	input signal for "POST_LATCH"
TIMESEL	input signal to ADIDAT used to change acquisition rate
TRIG	the trigger input line for module "CIRCUIT1"
TRIGGER	the trigger input signal into the ADIDAT
UPDOWN	used in conjunction with TIMESEL to increment/decrement rate
XCLK	the system clock input in "CLKS"
YD_NOT	an artificial trigger signal generated within the "MEM_CTR" module

CIRCUIT MODULES:

Note: This section is not listed alphabetically, but rather heirarchially

"FINAL1"	topmost module of the ADIDAT
"CLKS"	contained within "FINAL1"
"COUNT4F"	contained within "FINAL1"
"PRE_LATCH"	contained within "FINAL1"
"CIRCUIT1"	contained within "FINAL1"
"TIME_LATCH"	contained within "CLKS"
"PS2"	contained within "CLKS"
"PS4"	contained within "CLKS"
"PS10"	contained within "CLKS"
"MUX32TO1"	contained within "CLKS"
"UDC8"	contained within "CLKS"
"LATCH3"	contained within "PRE_LATCH"
"DMUX4216"	contained within "PRE_LATCH"
"LATCH5"	contained within "TIME_LATCH"
"DMUX4216"	contained within "TIME_LATCH"
"TESTER"	contained within "CIRCUIT1"
"POST_LATCH"	contained within "CIRCUIT1"
"RAM_LED"	contained within "CIRCUIT1"
"SELECTOR"	contained within "CIRCUIT1"
"MEM_CTR"	contained within "TESTER"
"PRETRIGGER"	contained within "TESTER"
"COUNT11F"	contained within "MEM_CTR" and "PRETRIGGER"

"MUX32"	contained within "MEM_CTR" and "PRETRIGGER"
"SEL"	contained within "PRETRIGGER"
"COUNT4F"	contained within "COUNT11F"
"COUNT2F"	contained within "COUNT11F"
"COUNT1F"	contained within "COUNT11F"
"LATCH11"	contained within "POST_LATCH"
"DMUX4216"	contained within "POST_LATCH"
"RAM_ADD"	contained within "SELECTOR"
"DATA_REG"	contained within "SELECTOR"
"ADD_MUX"	contained within "SELECTOR"
"ADD_SEL"	contained within "SELECTOR"
"SR4"	contained within "SELECTOR"
"TRI_REG"	contained within "SELECTOR"

KEYPAD:

Note: This section is not listed alphabetically, but rather in the order of appearance

EXECUTE	keypad key used to initiate an action
CAP_DIS	keypad key to select capture or display mode
DPR	Dual-Port RAM
LOCK	keypad key used to halt storage into display RAM
RAM	Random Access Memory
HOLD	keypad key used to halt storage in storage RAM
REL	keypad key to allow storage in the memory
ROLL MODE	keypad key to select the roll mode
REFRESH MODE	keypad key to select the refresh mode
SINGLE CAPTURE	keypad key used for a single event capture
CONT. CAPTURE	keypad key used for a continual storage of data
RECORD LENGTH	the amount of data to be stored in memory
1K	selection key for 1 kilobyte of data to be stored
2K	selection key for 2 kilobytes of data to be stored
LONG_REC	keypad key allowing a 16-Kbyte record
RESET_MODE	keypad key to change capture modes
PRETRIGGER	determines the amount of data stored prior to triggering
TIME UP	keypad key to step up the rate of acquisition
TIME DOWN	keypad key to step down the rate of acquisition
STORE	keypad key used in storing variables
RECALL	keypad key used in recalling a data record
CLR	keypad key used to clear stored variables
BLK_LD	keypad key to store the BLK variable

1. Introduction

In recent years, phenomenal growths and expansions have occurred in digital system techniques and applications. A trend is now prevalent where even traditional analog applications rely on some underlying digital technology. This innovation stems mainly because of the economic feasibility and data manipulation capabilities associated with digital systems. A typical example of the advancement into the realm of digitalization can be seen with the oscilloscope. Previously thought of as a purely analog testing device, the digital oscilloscope now easily outnumbers its' analog counterpart.

The progression into the digital domain from the analog necessitates the requirement of some type of converter which changes analog signals to digital ones. The method of conversion should be fast in order to maintain the resolution and accuracy of the original signal. The need for high-speed analog-to-digital converters is apparent, and associated with it are the resulting fast data streams which need to be handled. The emergence of fast digital data records can be seen everywhere in todays society, ranging from information sent by satellites to information transfer with the use of a FAX machine.

The electronics industry has made numerous technical advances in the design of high speed analog-to-digital converters [1-3]. This has been made possible through the increased use of VLSI design techniques and improved fabrication technologies. The resulting A-to-D component can contain a larger number of comparators for the same die size, and therefore be more accurate while remaining economically feasible. As principal modules in most data acquisition systems, this development has had a large impact on conversion performance of analog signals into digital data for processing purposes.

The rapid conversion of high frequency signals from the analog to the digital domain has not been the only challenge to overcome. The fast digital signal which results must then be processed by the system. Although digital signal processing (DSP) integrated circuits are widely available on the market, their processing capabilities are limited to signals in real-time applications in the audio frequency range. However, it is fortunate that not all situations require real-time processsing, that is, a system which performs instantaneous and continuous on-line analysis of incoming data. Many DSP applications can collect data, store it, and perform the processing task in subsequent off-line steps. In such quasi real-time systems, the data processing stage is not critically related to the timing of the data source, and thus, can be much slower. Although the analyzing step occurs at a reduced rate in comparison with a full real-time system, these systems still require fast and large memories to capture the required data quickly from the analog-to-digital converter [4].

With these requirements in mind, research was conducted in the development of a quasi real-time based, high speed data acquisition device. In terms of the present technology available, many design routes could be considered. However, keeping in mind that the circuit architecture could reach in excess of 7000 logic gates (an equivalent of 28,000 transistors), computer graphic tools for VLSI (Very Large Scale Integration) design were utilized. Design-work therefore involved the use of computer-aided design software, which is now quite commonplace in the microelectronics industry. The circuit can therefore be realized in the form of a single chip integrated circuit using a type of cell-based design architecture. With this scheme, a designer can "build" the circuit on a computer by selecting a pre-designed selection of logic gates and functions from a macrocell library. This library of parts allows direct cell selection and thus, the user need not

provide the interconnections for each individual component transistor [5,6]. This method of computer-aided design (CAD), called schematic capturing, not only provides the cell interconnections to build the entire circuit, but generally allows the user to simulate the result using available simulation software packages. Thus, the single chip design, referred to as *ADIDAT* (Analog-to-Digital Interface for Data Acquisition and Transfer) becomes a highly integrated solution which will allow the user to interface between a high speed A-to-D converter and a relatively slow microprocessor/RAM system in quasi real-time applications. Therefore, the ADIDAT, in conjunction with a temporary storage buffer, can be used in situations where the microprocessor cannot handle the fast A/D samples in real time.

The design of this application-specific integrated circuit, or ASIC, was brought about from the growing need to interface between fast digital data sources and relatively slower microprocessor-based systems. The acquisition device should not only manage the data systematically, but should do so in a cost-effective manner. To this end, the ADIDAT design described within this document, fulfills these requirements. This single chip design could be incorporated in many digital signal processing front-end applications, such as video sampling and image processing. However, for the purpose of this project, the aim of the design will be to implement it as a building block for the use in a digital storage oscilloscope (DSO).

The remainder of this discussion will deal with data acquisitions of digitized waveforms pertaining to the storage and retrieval methods found in digital storage oscilloscopes. System operation will be controlled by the dedicated circuitry found within the custom-designed integrated circuit, to be discussed in detail. The use of the ADIDAT in DSO applications is ideal because of the quasi real-time nature of the system. By using the device as a peripheral unit to an existing analog

oscilloscope, a hybrid analog/digital oscilloscope can be constructed with minimal external circuitry. This would require that the captured digitized data be re-constructed back into the original analog signal so that it may be displayed on a conventional oscilloscope screen. However, if the ADIDAT was used solely in a non-hybrid, digital oscilloscope format with a standard monitor, then the digitized waveform need not be converted back to the analog domain if a CRT (cathode ray tube) controller or video processor chip was employed. Since the final instrument makes use of a custom designed integrated circuit, the resulting component count will be drastically reduced with the benefit of reliability and ease of maintenance [7]. Thus, employing the ADIDAT would be a cheap and effective method of obtaining the characteristics desired from a digital storage oscilloscope.

The digital method of storing waveforms provides many significant advantages. Notably, it facilitates the viewing of pretrigger information and also allows the display of waveforms free from flicker or jitter over extended periods of time [3]. This report will provide a detailed examination on the operation of the custom designed gate array when used as a data acquisition device for a storage oscilloscope. As a preview to this, DSO concepts and operation will be discussed first.

1.1 Background

1.1.1 Concepts of an Analog Oscilloscope

As an introduction to oscilloscopes, the concepts of the conventional analog variety will first be analyzed. This will help in the understanding of basic oscilloscope operation. The cathode-ray tube, or CRT, is the indicating device for the analog oscilloscope. Within this vacuum tube is an electron gun which generates a beam of electrons to be directed towards the desired position on the

fluorescent screen. The electrons are positioned by vertical and horizontal deflection plates, as shown in Figure 1.1. If voltages are applied to both sets of plates simultaneously, the position of the electron beam in both dimensions of the plane of the CRT screen will depend on the sign and magnitude of the two deflection voltages. In this manner, the electron beam can produce an x-y plot of one deflection voltage as a function of the other [8]. Generally, the input signal voltage being measured by the oscilloscope isn't large enough to produce electron deflections, thus, deflection amplifiers are required to increase the signal voltage.

Prior to the deflection plates, circuitry is required in order to generate the two deflection voltages. The vertical signal (or y-axis signal) arises from the magnitude of the measured signal. The horizontal, or x-axis signal, usually corresponds to a time measurement of the input signal. The vertical signal is then applied to the vertical deflection amplifiers while the horizontal signal, or sweep, is generated by an internal ramp generator, giving deflection proportional to time [9]. A simplified block diagram for a dual-trace triggered analog oscilloscope has been given in Figure 1.1. From the diagram, the major circuit blocks can be seen. The two input channels feed into separate calibrated gain control amplifiers which determine the scale of volts/division on the screen. However, before reaching the gain control amplifiers, which includes vertical positioning and invert control, the signal passes through the selectable signal coupling. The oscillator circuit that produces the horizontal sweep signal is called the sweep, or ramp generator. The rate of change of ramp voltage must be very uniform in order to obtain a reliable time base. Good linearity of the sweep is a major requirement for a reliable oscilloscope [8]. Now that both vertical and horizontal signals are available, a voltage versus time graph can be produced. But to provide a continuous, stable display of a repetitive waveform on the screen, the starting time of the sweep must

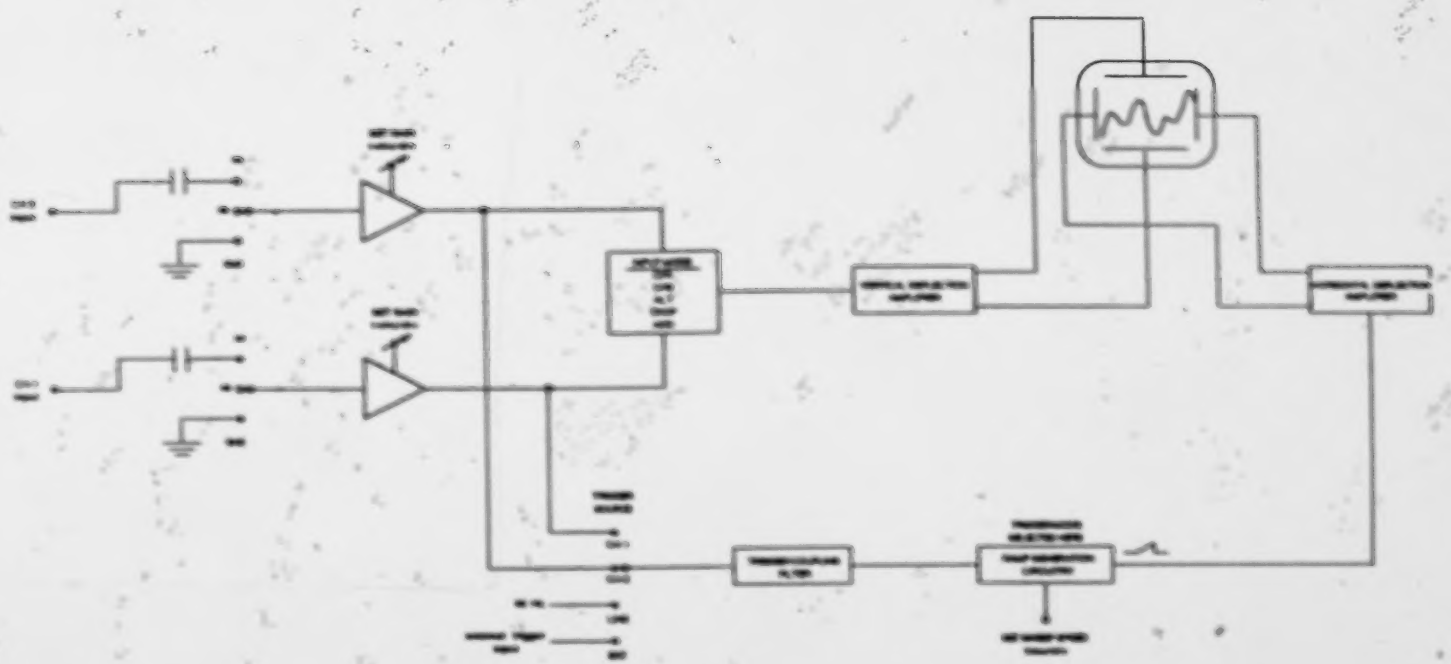


Figure 1.1

A Simplified Block Diagram of an Analog Oscilloscope

coincide with a single point on the waveform of the signal to be displayed. Triggering involves using the input signal to start the ramping process at exactly the same level and slope on the waveform for each sweep. With this, various trigger modes and coupling is possible. However, since a trigger initiates the display of a waveform on an analog oscilloscope, no information about the signal is available to the user prior to the triggering event.

In comparing an analog oscilloscope with a digital oscilloscope, many differences arise immediately. A DSO captures and holds the digitized waveform for long-term display on a CRT or monitor. Analog oscilloscopes use special screen phosphors or a charged mesh behind the screen to maintain the luminance of the display. With digital storage oscilloscopes, the data is digital in nature and can be stored or displayed for an indefinate period of time without any waveform fading or blooming. DSO's also allow for data transfer between the instrument and external recorders or computers. One drawback, however, is that because of the limited conversion speed of A/D converters, the digital storage oscilloscope bandwidth is not as high as that for its analog counterpart. This means that the DSO cannot record as high a frequency signal as an analog oscilloscope would be able to measure. Equivalent time sampling methods and faster A-to-D conversion techniques are allowing this restriction to be somewhat relaxed.

1.1.2 Concepts of the Digital Storage Oscilloscope

One of the key differences between analog and digital oscilloscopes is that the latter converts an analog input signal into a digital format. This representation of the analog signal can be extremely accurate if both voltage and time coordinates of each point are precisely known [2]. Retrieval of the original signal can be accomplished by digital-to-analog conversion, where the digitized waveform is

converted back to the analog domain for display on a conventional oscilloscope screen. As an alternate and easier method for re-display, the waveform can be kept in digital format for display on a monitor, providing that some type of video processing chip is used. With this technique, the need for a digital-to-analog converter can be eliminated, thereby further reducing possible conversion errors which may arise during re-construction. Thus, in comparison with analog oscilloscopes, digital storage scopes have the ability to capture and display transient or repetitive waveforms, but more importantly, they can store the digitized records in memory because of the digital nature of the signal [3].

With digitization, the number of binary words which can be captured in a given amount of time is governed by the frequency of an accurate crystal clock signal. Generally, a scaled version of the clock frequency drives a binary counter which has the ability to step through all the address locations in memory. This provides a simple method by which each sampled word is placed sequentially into memory. The period of the counter clock determines the horizontal resolution of the stored waveform. The vertical resolution of the stored signal is determined by the number of bits produced by the analog-to-digital converter [1]. Converters used in digital oscilloscopes are usually 6, 8, or 12 bit devices, however, most tend to use the standard 8-bit converter [10,11]. To achieve a respectable bandwidth rating, the A-D converter should be capable of high speed operation in order to satisfy Nyquist's well-known sampling theorem. For example, as a minimum requirement, an oscilloscope with a bandwidth of 10MHz must use a digitizing sampling rate of at least 20 MHz.

It should be noted that since this application has the characteristics of a quasi real-time system, the data processing stage can occur at a later time and at a reduced rate. For re-displaying purposes, the stored waveform only needs to be

clocked out of memory fast enough to prevent screen flicker, therefore the output devices need not be high speed.

Another major advantage, and probably the most important aspect of the digitizing oscilloscope is its ability to capture an event which occurs before the trigger point [1,12]. Recall that in analog oscilloscopes, the trigger initiates the horizontal sweep in order to produce a steady, readable waveform. Therefore, it is impossible to view events which have occurred prior to triggering using a standard analog oscilloscope. Pretrigger capture also makes it possible to view the initial portion of a transient or single-shot waveform, which an analog instrument would normally be unable to do. Figure 1.2 illustrates the concepts relating to pretriggering [10]. The digital circuitry utilized in this design is quite unique in facilitating pretrigger data capture. Before triggering occurs, the digital storage oscilloscope's memory is being constantly updated with the current input signal. However, upon triggering, the memory is either frozen if no posttrigger information was desired, or is otherwise allowed to run for a set interval after triggering [10]. This allows one to see what happened before and after the triggering point, depending on the setting of the pretrigger value.

Microprocessors are increasingly becoming standard components in digital storage oscilloscopes allowing further processing or mathematical manipulations of the digitized waveform. Programmability also allows keypad inputs, pre-setting of commonly used settings, as well as on-screen cursor control, which are all essential facets of oscilloscope operation.

Finally, because the input analog signal is in a digitized format, this data may be easily transmitted to other systems or supply data to a printer/plotter interface [1].

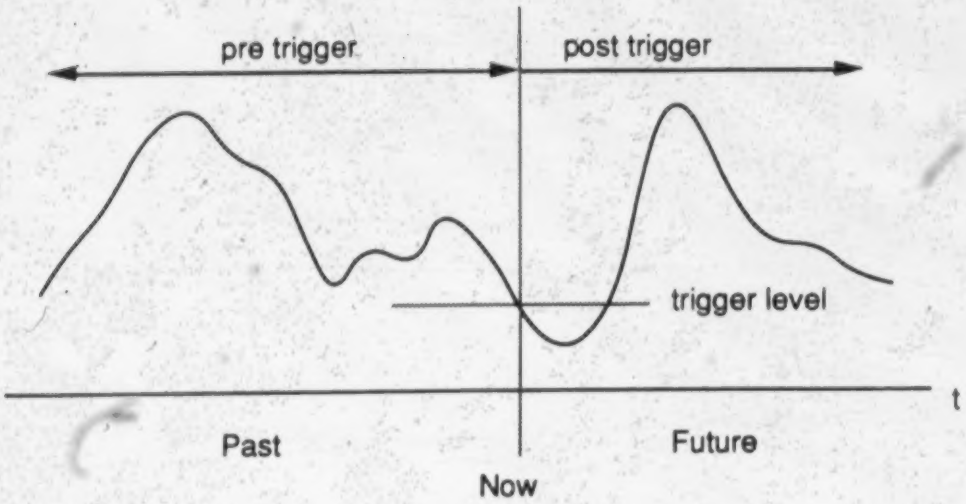


Figure 1.2

An Example Showing Pretrigger Information

For analog oscilloscope users, experiencing the advantages of a digital storage oscilloscope would require an entirely new instrument. However, with the research design proposed here, the external circuitry would be able to work in conjunction with an existing analog oscilloscope. This hybrid would therefore employ all of the standard components of the analog version, but also utilize A-D/D-A conversion circuitry, memory, and the custom designed logic array [3].

1.1.3 Introduction to Sampling

In a digitizing oscilloscope, the input waveform must be quantized into discrete time and voltage samples with the use of a fast analog-to-digital converter. There are basically two figures of merit employed in the comparison of

digital storage oscilloscopes, namely the bandwidth and the sampling rate. The digitizing rate is determined by the speed of the A/D converter, while the bandwidth can be related to the type of input signal being recorded. Much confusion arises when attempting to determine a relationship between these two performance parameters [13].

Consider an example of measuring a signal with a relatively fast transition, as shown below. When the samples are not spaced closely enough, it is impossible to determine the exact location of the transition edge, nor can its shape be fully ascertained. The effect is the same as using an analog oscilloscope with insufficient bandwidth. If the oscilloscope has a higher sampling rate as shown in Figure 1.3, the edge can be correctly located.

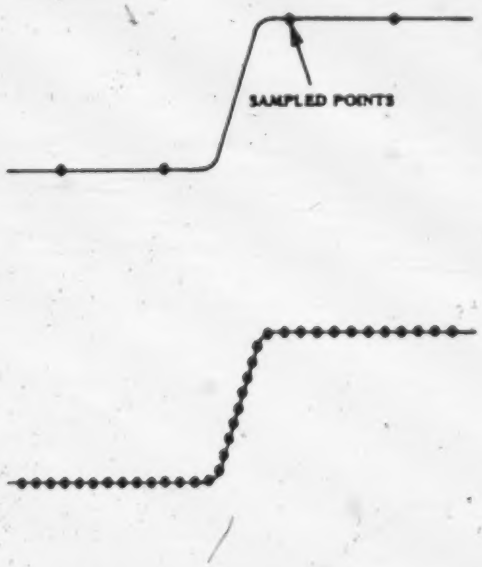


Figure 1.3

Signal Measurement Using Varying Sampling Rates

Recall that Nyquist's theorem states that if a signal is sampled at at a frequency of 2f, then there is no information in the samples about the components of that signal at frequencies above f. But this should not be interpreted to be read as the bandwidth of a real-time digitizing oscilloscope is half the sampling rate. The Nyquist limit is an absolute upper limit. Therefore, data cannot be transmitted as samples at a rate higher than the Nyquist rate, regardless of the location of the sampling instants, the nature of the set of frequencies which the signals occupy, or the methods of construction [14].

To illustrate why the bandwidth is not simply half the digitizing rate, consider a sine wave with a frequency f, where 2f is the sampling rate. In this situation, it is evident from Figure1.4 below, that it cannot be determined whether a signal is even present, much less what its amplitude is.

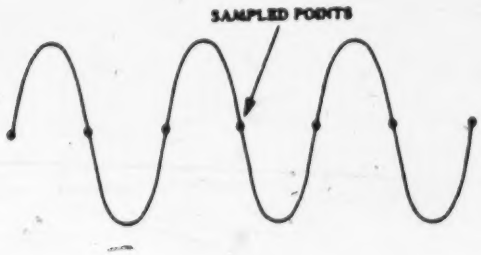
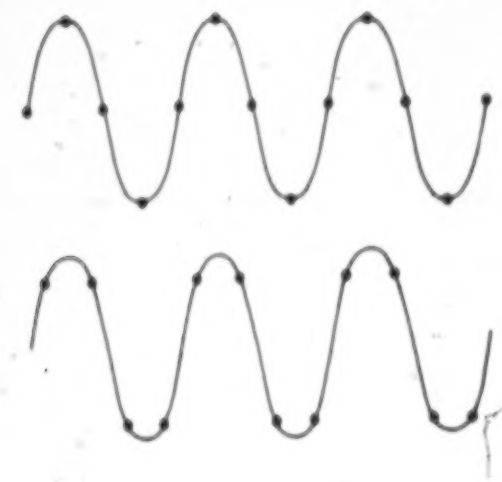


Figure 1.4

Sampling at the Nyquist Rate

To further illustrate this point, consider the situation where the sampling rate is four times the sine wave frequency, as shown in the Figure 1.5. In either of

the cases shown, signal regeneration is possible if the reconstruction algorithm included some assumptions about the shape of the original signal.



Two possible waveforms generated.

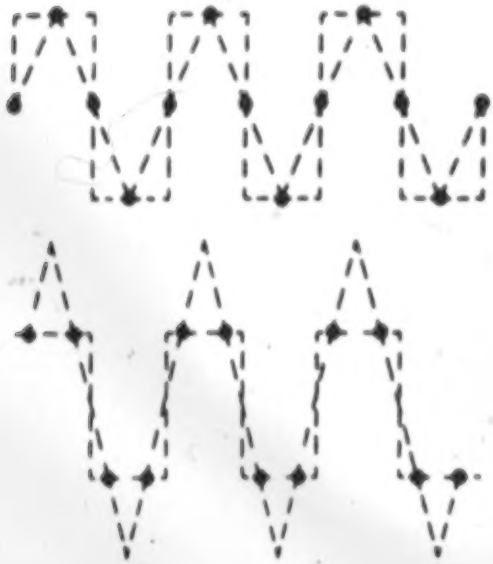


Figure 1.5

Sampling at Twice the Nyquist Rate

With either of the sampled points from the earlier figure, re-construction could have yielded a square wave, a triangle wave, or any other arbitrary waveshape [10]. The important point to note is that if a sine wave was reconstructed from the sampled waveform, then there was as much information about the original signal as if it had been passed through a band-limiting filter which removed all frequencies except the fundamental. This is equivalent to viewing the signal with an oscilloscope that has a bandwidth equal to f.

As a general rule of thumb, if the sampling rate is ten times the signal frequency, as in the Figure 1.6, an excellent reconstruction is possible by merely connecting the samples with straight vectors. Using this sample density, the eye can interpret the original signal visually with little or no error.

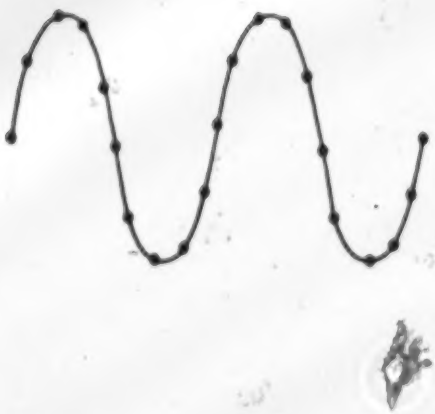


Figure 1.6

Sampling at an Adequate Rate

1.1.4 Flash Analog-to-Digital Converters

As stated previously, the digital oscilloscope must be able to sample an input analog waveform at a high rate so that distortion effects are limited upon regeneration of the original signal. By the Nyquist sampling theorem, an input signal should be sampled at a minimum of twice the highest frequency component of the waveform so that the samples exactly describe the original signal. In order to ensure accurate sampling, the input signal could be band-limited by an appropriate input filter prior to the sampling step. An anti-aliasing filter can be used to eliminate frequencies above the Nyquist limit, although this removes any indication to the user that higher frequencies are present in the input signal [15].

In order to ensure adequate sampling, the digitizers used in digital oscilloscopes tend to have a considerably higher sampling rate than that stipulated by the Nyquist criterion. It is therefore essential to have an extremely fast analog-to-digital converter within the oscilloscope circuitry to allow for the display of analog signals with large bandwidth.

The fastest analog-to-digital converter is known as the "flash", or parallel type converter. The high speed of the flash converter results because the analog signal is simultaneously compared with a reference voltage through a bank of parallel comparators. Speeds can range upto 200 MHz, while most commercial devices reach up to 20-40 MHz. Although the fastest designs utilize ECL (emitter-coupled logic) connected bipolar transistors, CMOS conversion rates can also attain the megahertz range [16].

The flash architecture consists of a set of 2^n-1 comparators which perform the quantizing operation for n-bit conversion. The comparators all have different

threshold values set by the reference voltage and a resistive divider network. By comparing the input analog signal with a specific reference voltage, and depending on which input value is greater, the comparator produces a logic "0" or logic "1" output accordingly. The result from the comparators is a set of 2^n-1 bits which need to be further converted into a normal binary code. Generally, some type of decoder logic is required to produce an acceptable digital code, although in normal circumstances, the decoding is done within the A-to-D converter itself.

Despite their high speed of operation, parallel converters do have some unfortunate disadvantages. Flash converters generally tend to be limited to eight bits because for greater resolution, the circuit complexity approximately doubles for each additional 1-bit increase [17]. Another limit to a flash converter is that at high frequencies, aperture delay effects become more predominant. Aperture delay time is the interval between a sample signal and a comparator decision. This can become a serious source of distortion if each comparator's aperture delay time is different.

Many analog-to-digital converters require a sample-and-hold circuit to process the signal before it is applied to the converter. This is to ensure that the voltage level of the input analog signal is stable during A-to-D conversion. Even if a sample-and-hold is not required, interfacing an analog signal to a flash converter can be quite difficult. Since the input capacitance and resistance of the A-to-D varies with signal level, it is important to drive these fast converters from low-impedance sources, such as op-amps or buffers.

For this project, the flash analog-to-digital converter being considered is Samsung's KSV3100A-8 data conversion IC, which can operate at speeds of up to 20 MHz. This chip consists of a high-speed flash-type 8-bit A/D converter and a

high-speed low-glitch 10-bit D/A converter designed as an R-2R network. Along with the basic conversion circuitry, a major advantage of this IC is that it contains auxilliary circuits which provide versatility in potential applications needing a minimum of external components. For example, an impedance converter is connected before the A/D to provide a high-impedance input. Also, a reference voltage source for the A/D is generated on-chip. Other circuitry which has been integrated onto this single IC is an input clamping circuit and a feed-in output amplifier. Along with these advantages, SAMSUNG is currently designing a 40 MHz A/D-D/A integrated circuit which would be chip-compatible with the KSV3100A-8. The block diagram for the KSV3100A-8 is shown in Figure 1.7, and a diagram showing the major pin-outs for this I.C. is given in Figure 1.8 [18].

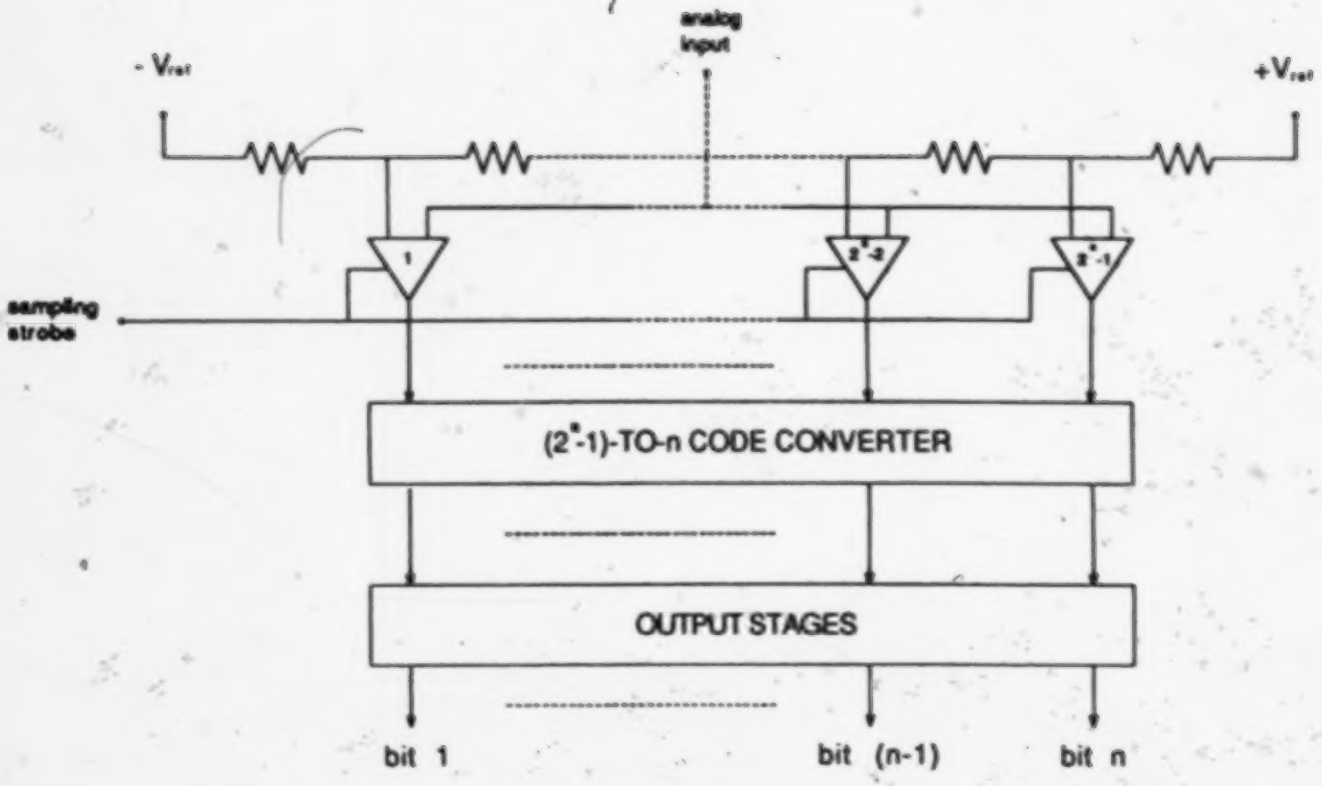


Figure 1.7

Block Diagram of the Flash ADC

THE MATERIAL CONTAINED ON THIS PAGE
HAS BEEN OMITTED DUE TO THE UNAVAILABILITY
OF COPYRIGHT PERMISSION.

This figure taken from "Linear IC Data Book, Vol. 2"
as indicated in reference 18.

Figure 1.8

Pinout for Samsung's KSV3100A-8 ADC

2. Designing the ADIDAT

2.1 Previous Work

In the context of an ADIDAT-based digital storage oscilloscope, the initial design-work had to start from the very basics since previous works consist only of the finished product, namely a digital oscilloscope. These oscilloscopes can possess a varied number of characteristics, features, and price ranges. In order to gain an understanding of what features and technologies are available in the market, it would be beneficial to familiarize ourselves with the basic digital storage oscilloscope concepts and operation. By determining the characteristics which are available in the marketplace today, a strategy of features can be developed for the ADIDAT-based DSO. These have been fully discussed in the section entitled "System Overview".

It must be kept in mind, however, that the purpose of this project is to design a single-chip ASIC which will handle much of the data management in accordance with the basic features desired in a digital oscilloscope. As an additional benefit, the reliability of the fabricated circuit increases while costs are reduced since only a single integrated circuit is involved. The ADIDAT design will therefore allow upgrading to the digital technology level for analog oscilloscope users, and will do so in a cost-effective manner.

2.2 Design Considerations

The method by which the logic circuitry will be implemented is normally not a particularily important aspect to consider when preparing a design document. Generally, design-work is conducted with no regard to gate counts or the type of technology being used. However, since the goal of this thesis was to make

significant headway towards circuit realization, consideration has been given to examining the numerous implementation options which are available. With the final choice having been made, the design must take into account the technology-based restrictions particular to that implementation scheme.

During the conceptual phase of this design, a comparison was made between the ultimate use of discrete medium-scale integration (MSI) components, a fully-custom designed IC, or a semi-custom designed CMOS logic array.

With off-the-shelf discrete components, circuit construction time is long and circuit evaluation can properly occur only after the entire design has been built. Also, troubleshooting could pose a problem because of the large number of chips involved along with its associated PC board size and complexity. It is also apparent that with increased chip count comes increased cost. With a fully customized implementation, design time is long and complex, but this results in a dramatically reduced chip count thereby increasing circuit reliability. More importantly, troubleshooting and performance characteristics of the circuit are all conducted and determined during the actual logic design [7]. However, the costs incurred with a full custom design are enormous because of the number of fabrication masks required. On the other hand, with gate-arrays, all of the characteristics of a fully-custom designed IC are realizable, except for the cost over-runs and the turn-around time in receiving the final product. The only cost involved is in the manufacturing of the metallization masks, as opposed to all of the process masks for a full custom design. The problem with any type of customized chip is that once it has been fabricated, the circuit cannot be changed or modified in any manner. As an alternative, the logic design may be carried out on another type of custom IC called an erasable programmable logic device, or EPLD chip. As the name suggests, the design is erasable so that any changes can be re-

programmed onto the chip. With EPLD's, the chip count may increase slightly in comparison with a single-chip gate array design because of the limited number of gates available on a single EPLD. As a result, the designed circuit must be partitioned in a modular format to allow for easy interconnections between EPLD's [6].

After carefully considering all of the options and noting the available resources, it was decided that the semi-customized gate-array approach best suited the requirements for this project.

2.3 Design Procedure

The formulation of complex designs such as the ADIDAT, involves many steps in deriving a viable solution. However, to assist the designer in finding a solution, a model for design representation and synthesis can be followed to systematically represent the entire design synthesis process. The model which has been presented [22], separates the behavioural, structural, and physical information about the design into three domains. Furthermore, each domain has been hierarchially partitioned to define levels of design abstraction. These levels of detail are labelled architectural, algorithmic, functional, logic, and circuit. A Y-chart has been used to describe the domains of description from an axial viewpoint. This has been shown in Figure 2.1 on the following page [22], where each of the three axes defines a particular domain. The level of abstraction decreases as one moves towards the vertex.

From the Y-chart, design synthesis can be defined as the process of translating a high level behavioural domain abstraction to a low level physical domain description. This would involve building inter-domain links from the behavioural to structural to physical domains. The starting point for any design is

to describe the behaviour of the proposed circuit. This action can be represented by an entry arc which is drawn on the behavioural axis of the Y-chart. From this point, movement is made by advancing towards the vertex in the same domain, or maintaining the same level of abstraction in a different domain. Progression will ultimately lead to a physical domain description.

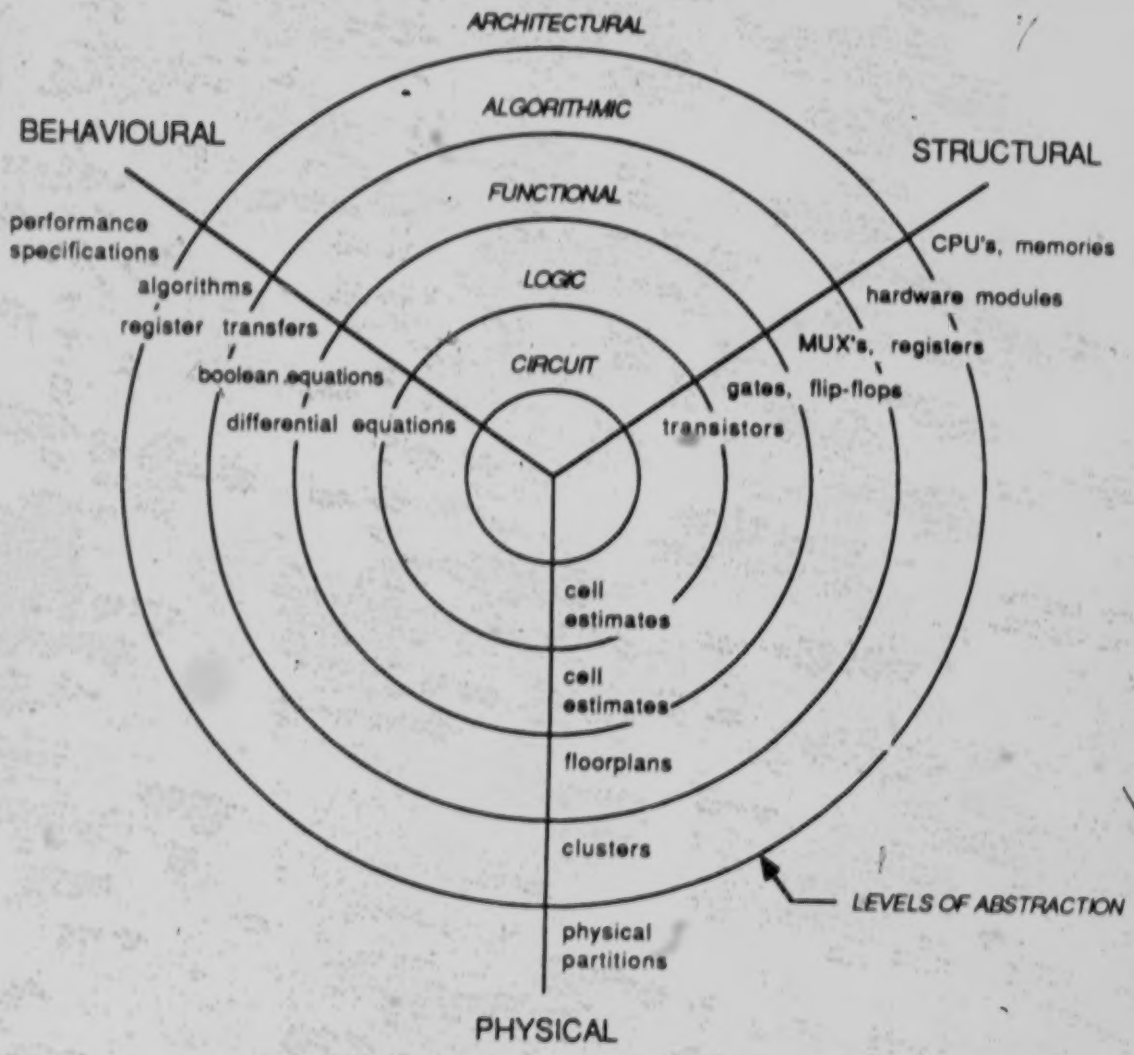


Figure 2.1

The Basic Y-Chart

Each of the three domains may also contain both a description component and a constraint component. A set of constraints attribute limitations to a design, for example timing and fan-out. The constraints which can be imposed in the physical domain may be thought of as the design simulation process.

This methodology of circuit synthesis was incorporated while designing the ADIDAT with the schematic capture system. Design-work was confined mainly to the behavioural and structural domains. This is the case in most VLSI design environments as the physical description need not be considered by the user since the end result is a single chip IC. The software package handles the internal cell partitioning, floorplanning, and routing once the entire design has been entered. However, the simulation constraint in the physical domain was involved in the design process for all cells in the design hierarchy.

As an illustrative example, a certain cell in the design required a method by which a 2-Kbyte RAM module had to be sequentially addressed. Since sequential addressing is most easily achieved by the use of a counter, an 11-bit binary counter was intuitively employed to access the 2048 memory locations. These steps correspond to an entry arc, an arc towards the vertex of the behavioural domain, and an arc at the same level of partition in the structural domain. Furthermore, the 11-bit counter has to be made up of smaller sub-counters, which are in-turn, formed by registers and individual flip-flops. Finally, as a result of the CAD software, the 11-bit counter is partitioned in the physical domain into smaller logic cell estimations, which in turn produces further circuit partitioning. The entire process for this cell is shown on the Y-chart of Figure 2.2 [22].

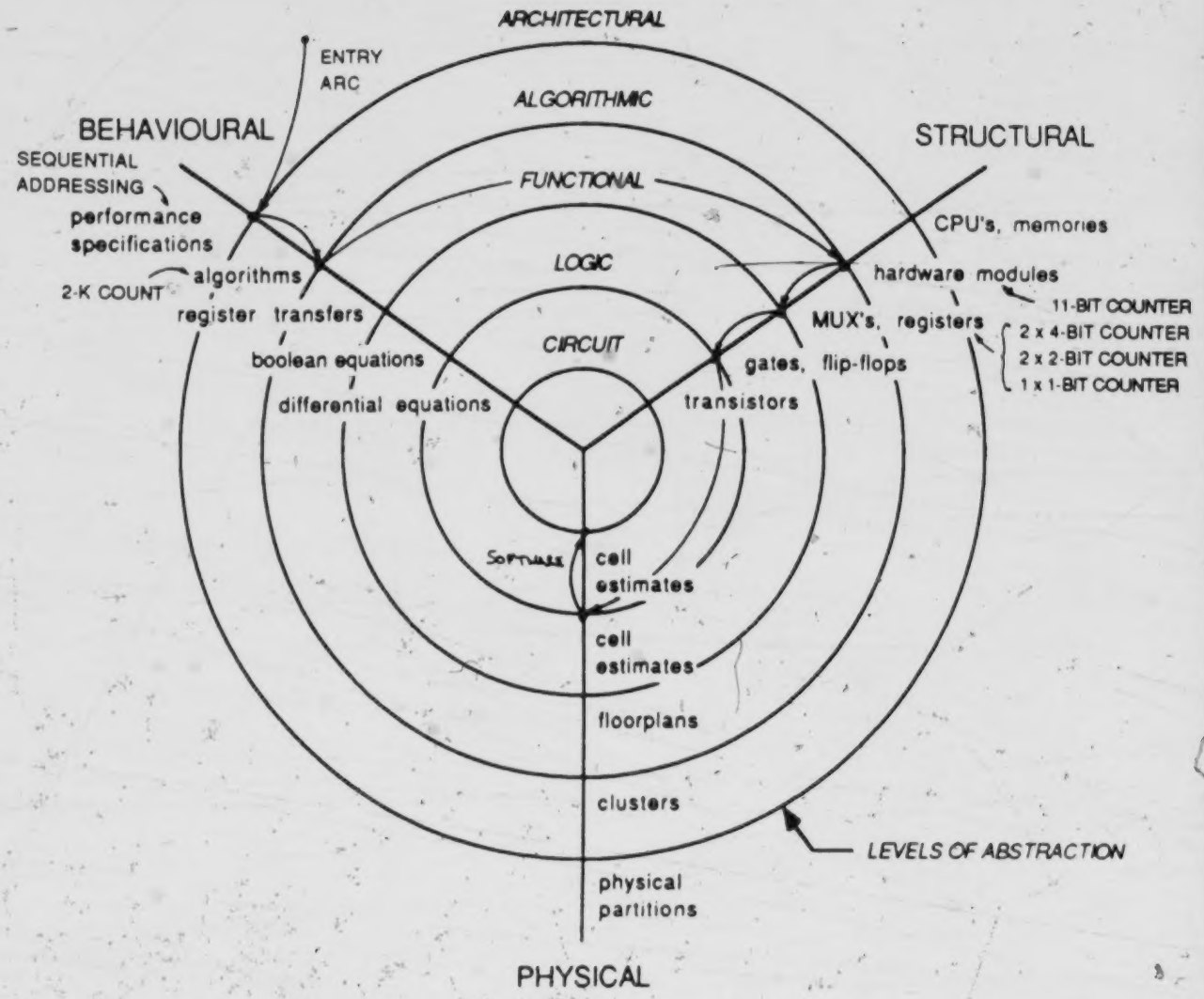


Figure 2.2

The Y-Chart Used in the Design Process

2.4 System Overview

As mentioned earlier, the logic circuitry contained within the custom-designed ADIDAT chip allows for its use in numerous data acquisition applications. But by the nature of this design, the circuitry has the ability to not only store data on a valid trigger level, but it also can capture a predetermined amount of data prior to triggering. Thus, with this type of pretriggering capability, the ADIDAT is especially well-suited in being utilized as a data acquisition unit in the architecture of a digital storage oscilloscope. This application was the original impetus for the design of the ADIDAT control chip.

Many digital oscilloscopes are available on the market today, each with its own unique features and characteristics. As with any electronic test instrument, the price tends to be directly proportional to the number of features desired in a device used for electrical analysis. Thus, the need for a low-cost, high-speed storage oscilloscope with the ability to store large data records is understandable. By using the ADIDAT in conjunction with an existing analog oscilloscope, this need can be fulfilled. With the internal circuitry of the ADIDAT module, many major design objectives were met. The primary capability was the ADIDAT's ability to store pretrigger data, which was briefly discussed earlier. Another objective was to allow the user a greater amount of memory and greater control in partitioning the memory. Reasonably priced digital storage oscilloscopes on the market today have about 4-Kbytes of memory available. With the use of the ADIDAT's memory multiplexing scheme, the user has 16-Kbytes of memory to store varying data records. This has the advantage of being able to store more information because of the larger memory available, but also allows the system to use slower and inexpensive static RAM. Sixteen 1-Kbyte records, eight 2-Kbyte

records, or a combination of 1- and 2-Kbyte records are possible. An additional option allows the user to select a "long record" storage, meaning the data record length will be stored within all of the 16-Kbytes of memory. Another design criterion was to be able to use high speed analog-to-digital converters while retaining an 8-bit accuracy. The most likely candidate for this task would be an 8-bit flash converter, which provides a compromise between cost, speed, dynamic range, and memory size. Finally, the reason for designing an application specific integrated circuit (ASIC) using gate array technology was mainly to reduce the chip count. This tends to increase the reliability of the final product and is easy to package. As well, the design software allows testing and troubleshooting of the circuit during the logic design phase.

2.5 The ADIDAT-Based DSO:

A system overview will now be given with the ADIDAT forming the principle module in a digital storage oscilloscope design. The following block diagrams will be useful aids in this discussion. Figure 2.3 illustrates the basic block diagram for a DSO [10,8] while Figure 2.4 shows how the ADIDAT would fit into a general circuit diagram for a digital storage oscilloscope.

2.6 Digital Storage Oscilloscope Operation

The design of the ADIDAT was accomplished with certain digital storage oscilloscope operations in mind. However, not all DSO features can be implemented in this gate-array design since the technology restricts us to a maximum of 10000 gates and 224 pins per chip package. Automatic routing problems will further limit the effective gate count to 8500 [19]. Thus, with these

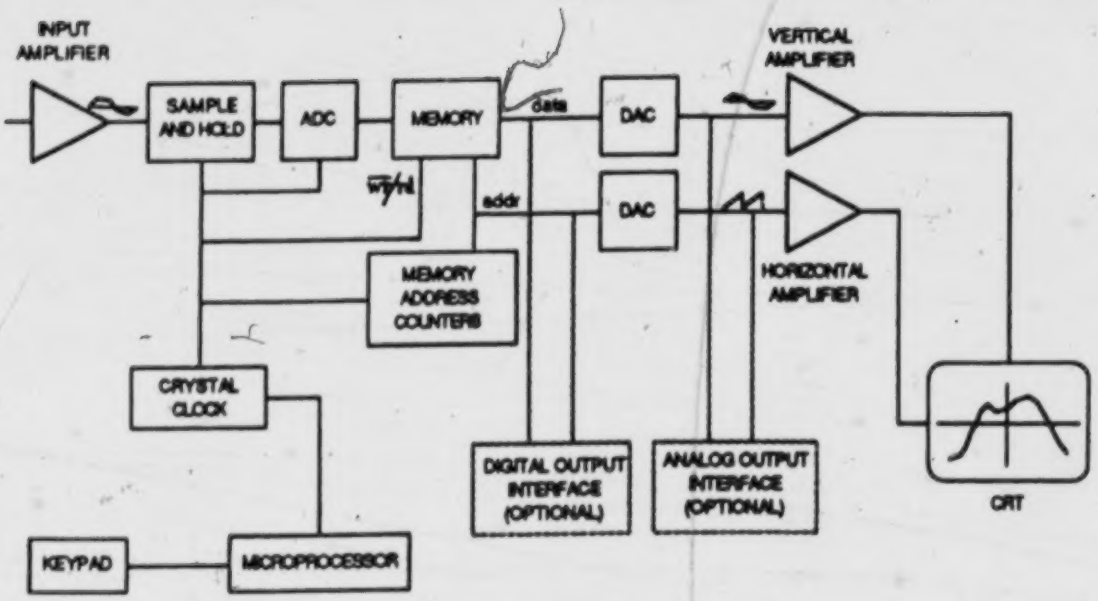


Figure 2.3

A General Block Diagram for a Digital Storage Oscilloscope

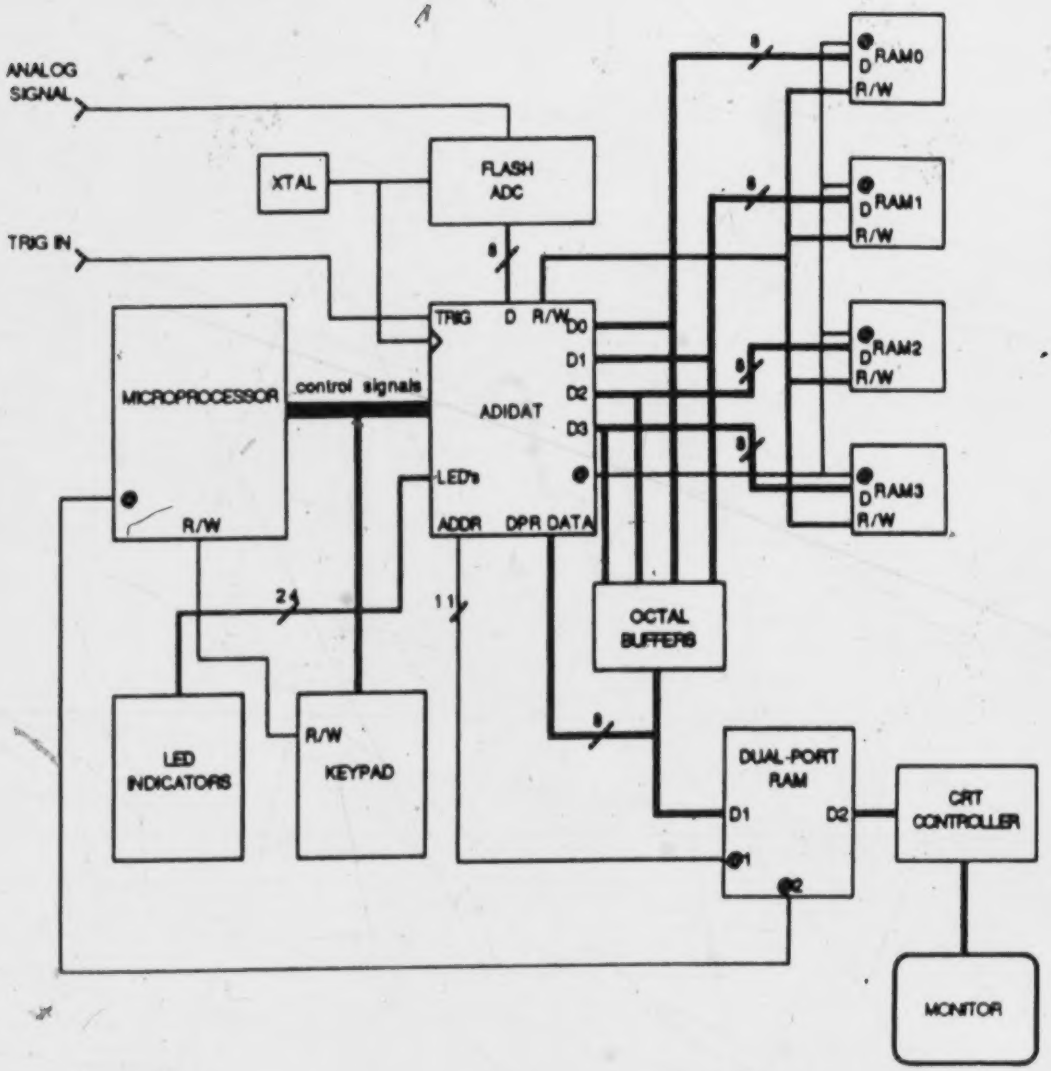


Figure 2.4

The DSO Using the ADIDAT Module

restrictions, only the primary DSO functions are realized in the design framework. The following sections provide a detailed look at the available characteristcs [20].

2.6.1 CAPTURE CONTROL:

Since the ADIDAT is primarily a synchronous device, it is activated only when the external oscillator is enabled. When this occurs, the chip is ready for the acquisition and handling of incoming digital data. The method by which data is stored or displayed depends on the specifications entered by the user.

There are two storage modes available with the ADIDAT. Either can be selected through the REFRESH or ROLL control keys found on the user keypad. Along with the mode of storage, the user must also determine the type of capturing method desired. By using the keypad, the user can choose between either a single capture acquisition or a continuous capture format. The storage modes, capturing techniques, and control variables will now be examined individually.

2.6.2 REFRESH:

Selection of the REFRESH mode produces a display similar to that of a conventional oscilloscope, with the exception that the trace is flicker-free even at low sweep rates. Operation in the single capture mode causes the instrument to store data only upon receipt of a valid trigger. The display updates from the left hand side of the screen to the right, with new display information over-writing the old. No pretrigger data can be captured in this mode of operation since a sweep is initiated only by a valid trigger. Thus, the updated display only contains information captured on or after the triggering event. If a trigger is not present, the displayed information captured on the previous sweep will be retained indefinitely

until the instrument is rearmed by the RESET_MODE key. This button, when activated, sets the device for single capture mode. However, if the continuous capture mode has already been chosen, then the captured waveform is displayed only until a new trigger is received, at which point the display begins updating again over-writing the previously stored waveform. This process can be halted by resetting the ADIDAT.

2.6.3 ROLL:

The selection of the ROLL storage mode with the continuous capture method provides a form of free-running trace not found on conventional oscilloscopes. The display should roll continuously, ignoring any incoming triggers, resulting in all sampled data being sequentially and continuously transferred into the 2-Kbyte display RAM. As the memory receives new data, the trace is moved to the left by one store location for each measured data point over and above the 2047th point. This is accomplished with the use of the dual-port RAM for the display memory. One port is accessed by the ADIDAT, while the other side is accessed by the system microprocessor. Displaying occurs simply by the microprocessor reading each of the RAM locations sequentially. The starting address is at the very top of RAM, but as each new data point over 2K enters the memory, the starting pointer is incremented by one. As the trace is updated, the display on the screen appears to be moving or "ROLL-ing" to the left, in appearance similar to a chart recorder output.

When the single capture mode option has been selected, pretrigger information can be obtained, and is explained as follows. At any particular moment, T, as data is continuously transferred into memory, the RAM contains only data captured before time T. If the storage process captures 255 points before

T and fills the 1-Kbyte record after T, then the display will contain 25% of the data captured before time T for a 1-Kbyte storage record. If T represents the time at which a valid trigger occurs, then a record with 25% pretrigger information has been captured in the case of a 1-Kbyte record. This is the basis of the pretrigger facility available with most digital storage oscilloscopes. The pretrigger options available will be explained next, and the unique digital circuitry providing this facility is explained in the section "PRETRIGGER CIRCUITRY".

2.6.4 PRETRIGGER OPTIONS:

Selection of one of the five pretrigger value buttons in ROLL mode, single capture option determines the amount of data displayed before the triggering point. Thus, the amount of pretrigger selected determines the percentage of the displayed data that is required before the arrival of the trigger. This is accomplished by the continual storage of incoming data until the triggering occurs. As an example, if a trace is to be captured with 25% pretrigger and is a 1-Kbyte record, incoming data is stored in the first 255 storage RAM locations immediately after the acquisition phase has been executed. After a valid trigger has been received, subsequent data will be stored in storage locations 256 to 1023, thereby preventing the pretrigger data from being over-written.

As mentioned earlier, five pretrigger options are available to the user, namely 0% meaning only posttrigger information is desired, 25%, 50%, 75%, and 100% pretrigger. The last option implies only data before the triggering event is needed. It should be noted that with the selection of the REFRESH mode, the pretrigger value is automatically set to a default value of 0%.

2.6.5 STORAGE RECORD SIZE:

In the earlier discussion of the design circuitry, there was an indication that a variable record size is possible. This is in fact true, and the one-bit input variable which accomplishes this is MEM_SEL, which specifies the data storage length to be captured. With MEM_SEL equal to 0, a 1-Kbyte record is stored while a logic 1 enables 2-Kbytes to be captured.

This variable controls the count length of the programmable 11-bit counter and is used to determine the final address value through multiplexing. This will be explained in the memory block selector section which follows, and within the actual circuit description.

2.6.6 MEMORY BLOCK SELECTOR:

The size of storage memory in a digital storage oscilloscope determines how much digital information can be stored within it [3]. Memory segmentation is generally used if more than one waveform is to be captured and saved [10]. The number of bytes that can be stored determines the horizontal resolution of the captured waveform, thus, memory partitioning should not be too small to cause a reduction in resolution. Similarily, partitioning should not be excessively large to cause memory waste. Using 1-Kbyte of RAM to store a waveform gives a horizontal resolution of 1023 steps which will give good reproduction of the original signal when the memory section is fully displayed [11]. Many digital scopes often use larger memories than 1-Kbyte so that more points are available to accomodate a zooming facility [10]. This aspect was considered when designing this chip, thus the user has an option of either selecting 1-Kbyte or 2-Kbytes of record length, depending on the value of MEM_SEL. This variable has the effect of enabling the

most significant bit of the 11-bit counters thereby increasing the address generating count from 1023 to 2047.

With the record length established, the user can now determine the starting location of the digitized waveform. The 16-Kbytes of memory available for data storage can be either partitioned into 1-Kbyte segments, 2-Kbyte segments, or a combination of both. Each record in memory starts either at the 1K or 2K boundary respectively, depending on the value of "MEM_SEL". Specification of the memory block is accomplished by setting the "BLK" variable, which has been separated through internal circuitry into "BLK3" (the 3 most-significant bits of BLK) and "BLSB" (the least-significant bit of "BLK"). The resulting 4-bit number allows the user to select blocks 0 to 15 for a 1-Kbyte record. If 2-Kbyte records are desired, then the block value is such that it is only dependent upon the "BLK3" variable, while the value of "BLSB" does not affect the situation. This is due to the concatenation of "BLK3" (3-bits) with the outputs of the 11-bit counter in the increased record situation. The "BLK3" value becomes the three most significant bits of the resulting 14-bit address. With 1-Kbyte of data points, concatenation occurs with "BLK" (4-bits) and the 10-bit counter outputs. The appended values for the block are controlled by a multiplexer selectable through the "MEM_SEL" input. Thus, with this design, only even numbered blocks from 0 to 14 can be chosen to accomodate a 2-Kbyte record.

To reiterate, the "BLK" variable is used as the 4 most significant bits of the destination address, and works in conjunction with the 11-bit counter. Thus, by specifying the "BLK" value, the base address has been set for a particular record. As described earlier, for 1K of data, the counter has 10 output bits plus 4 bits for block specification, resulting in a total of 14-bits. In the case of a 2K record, 11 output bits from the counter are present, but only 3 block specifier bits are needed.

Thus, for either situation, 16-Kbytes of memory is addressable with the 14-bits which are available. It is important to emphasize that the user is not restricted to only one record length of storage. Since a mixture of storage lengths is possible, it is necessary to prevent any over-writing of varying data lengths. A solution to this is by the use of an LED display indicator, which provides a visual aid in determining which storage RAM blocks have been filled. This is fully described in the section heading "CONTROL CIRCUITRY FOR THE EXTERNAL LED's".

2.6.7 LONG RECORD OPTION:

In special circumstances, a situation may require the user to store an input waveform for a longer period of time. The usual storage record length of either 1-Kbyte or 2-Kbytes may not be adequate for a certain application, so the user may select the long record storage option, which allows 16-Kbytes of storage for a single record. By chosing this selection from the keypad, the "LONG_REC" input line goes from a normally low state to a high logic level. However, this option must be used in conjunction with the continuous capture mode of operation. Upon execution, storage of the digitized signal is continuous with all incoming triggers disregarded. The signal is actually stored as 16 x 1-Kbyte records, and the acquisition phase is terminated after all 16-Kbytes of data have been fully captured by the memory.

2.6.8 MEMORY BLOCK LOADING:

Once the memory block for data storage or display has been determined by the user, it is necessary to hold the values of "BLK3" (3 most significant bits of BLK) and "BLSB" (least significant bit of BLK). The value for the memory block specifier must remain constant during the entire data acquisition and transfer

phases of either 1-Kbyte or 2-Kbyte records. However, for the case where 16-Kbytes of data need to be stored during a single acquisition, the "BLK" specifier does not remain constant. For the first Kbyte of data stored in this situation, the "BLK" value reamins at 0 (hex), but for each subsequent 1-Kbyte of data, the value of "BLK" should increment by one. This immediately suggests the use of a 4-bit counter which starts at 0 (hex) and increments to F (hex) for each clock pulse received. Thus, all the memory blocks are accessed by the 4-bit counter. The clocking occurs after a Kbyte of data has been stored, thus causing the counter to increment by one.

With this scheme, a 1-Kbyte record can be stored at a particular RAM location by setting the "BLK" variable and loading it into the counter via the preset lines. For 2-Kbyte records, an even numbered "BLK" value is selected and the reasoning behind has been explained in the earlier section. Loading is accomplished on the active low signal through the variable "BLKLOAD", which is taken momentarily low before operation of the ADIDAT is executed. The counter clock is disabled for these instances so that the "BLK" value remains constant and does not increment during a data handling sequence.

2.6.9 STORE, CLEAR, and RECALL:

In order to latch the user-input values of pretrigger percentage and timebase setting for a specific record, the "STORE" key is used. This line is active high and enables the latches corresponding to the four most significant bits of the base address determined by the memory block specifier. Once the line is returned to a low logic level, the data within the register is latched. The "STORE" line is used as input to both PRE_LATCH and TIME_LATCH, while storing in the

POST_LATCH (register which holds the count value upon trigger arrival) is accomplished by the logic circuitry.

The "CLEAR" key is used to clear a particular data latch, again specified by the memory block specifier. This line is also active high and causes the direct clear lines of the latches to be enabled. The "CLEAR" line is used as inputs to all three latches described earlier, namely PRE_LATCH, TIME_LATCH, and POST_LATCH.

The "RECALL" input is taken low when a data transfer from storage RAM to display RAM is to take place. This action allows the pretrigger counter to be loaded with the appropriate value stored in the POST_LATCH for a given record.

2.6.10 TIMEBASE SELECTION:

Any internally generated timebase can be selected through the "TIMESEL" input line, which effectively steps through each of the possible acquisition rates. By making use of the circuits' up/down counter, the stepping process can occur in either forward or reverse directions. Pressing the "TIME UP" key once advances the step by one to the next acquisition rate, while "TIME DOWN" steps down by one step. So once the desired timebase has been reached, and provided the user-defined pretrigger setting has been made, then the "STORE" key may be pressed. This stores the selected values in the appropriate latches.

2.6.11 EXECUTION CONTROL:

All data acquisition and transfer operations available with this device are initiated from the keypad by the "EXECUTE" button. Once a decision has been made to store or transfer data, the user must input the desired commands to obtain

the correct waveform record. During data acquisitions, the storage mode is specified, the choices being between the ROLL mode or the REFRESH mode. The destination or source area in RAM where the input data will reside or already exists, is then selected. The next set of commands deal with how to capture and display the input waveform. First, a specification must be made in regards to the amount of data to be stored, that is, the record length. The input signal "MEM_SEL" in FINAL1 determines whether the record will be 1-Kbyte or 2-Kbytes in length, and is selected through the keypad using the "RECORD LENGTH" buttons. Secondly, the amount of pretrigger information to be captured is selected, the choices being 0%, 25%, 50%, 75%, and 100%. With reference to FINAL1, these circuit inputs can be characterized by the table shown in Figure 2.5 on the following page.

To select the pretrigger value, the appropriate key on the keypad is pressed, after which the system will take line "STORE", in FINAL1, high. This causes the selected pretrigger value to be latched in "PRE_LATCH" and is assigned to the specific record for the chosen memory block. The "STORE" line is then returned to a low state. Also, once the pretrigger value has been established, the posttrigger counter preset's can be determined by the internal circuitry, and depend on which record length was selected. This is explained in the section dealing with waveform retrieval. After all acquisition or transfer commands have been entered by the user, the "EXECUTE START" key is depressed thereby enabling the internal clocking circuitry.

However, different actions are initiated by the address counters depending upon the storage mode selected (that is, either ROLL or REFRESH). If the storage mode is in ROLL, the counters begin counting immediately whereas in the REFRESH mode, the counters will start incrementing only when a valid trigger

PRETRIGGER	PRE100	PRE50	PRE25
0%	0	0	0
25%	0	0	1
50%	0	1	0
75%	0	1	1
100%	1	0	0

Figure 2.5

Pretrigger Options

has been received. Since the "EXECUTE START" line feeds the active-low "set" input of an SR flip-flop, the line goes low momentarily after being executed, then returns to its normally high level.

2.6.12 THE KEYPAD:

To facilitate in the ease of operating the ADIDAT-based digital storage oscilloscope, a custom-made keypad should be interfaced with the system microprocessor. The various input signals needed by the ADIDAT for proper operation would arise from the microprocessor, and their values would be determined by the keypad entries. A brief discussion on the function of each key follows, and an example of the required keypad has been given in Figure 2.6 on the following page. These pages should be used as a reference when reading the section entitled "Digital Storage Oscilloscope Operation".

In order to initiate any operation with the ADIDAT, the EXECUTE "START" key must be pressed after all acquisition or transfer commands have been entered. The "START" key is used to enable the internal clocking circuitry, while the EXECUTE "STOP" key is used to disable the the clocking circuit. Pressing "STOP" terminates any process which may be underway within the ADIDAT.

Most digital storage oscilloscopes allow the user to select the method by which data can be captured. The operaton of the ADIDAT module during a capture sequence depends not only on the capture method, but also on the MODE of data storage. The MODE of data storage is user-selectable with the aid of two keys labelled "ROLL" and "REFRESH". Similarily, the CAPTURE mechanism can be chosen, where the single-shot technique can be selected by pressing the

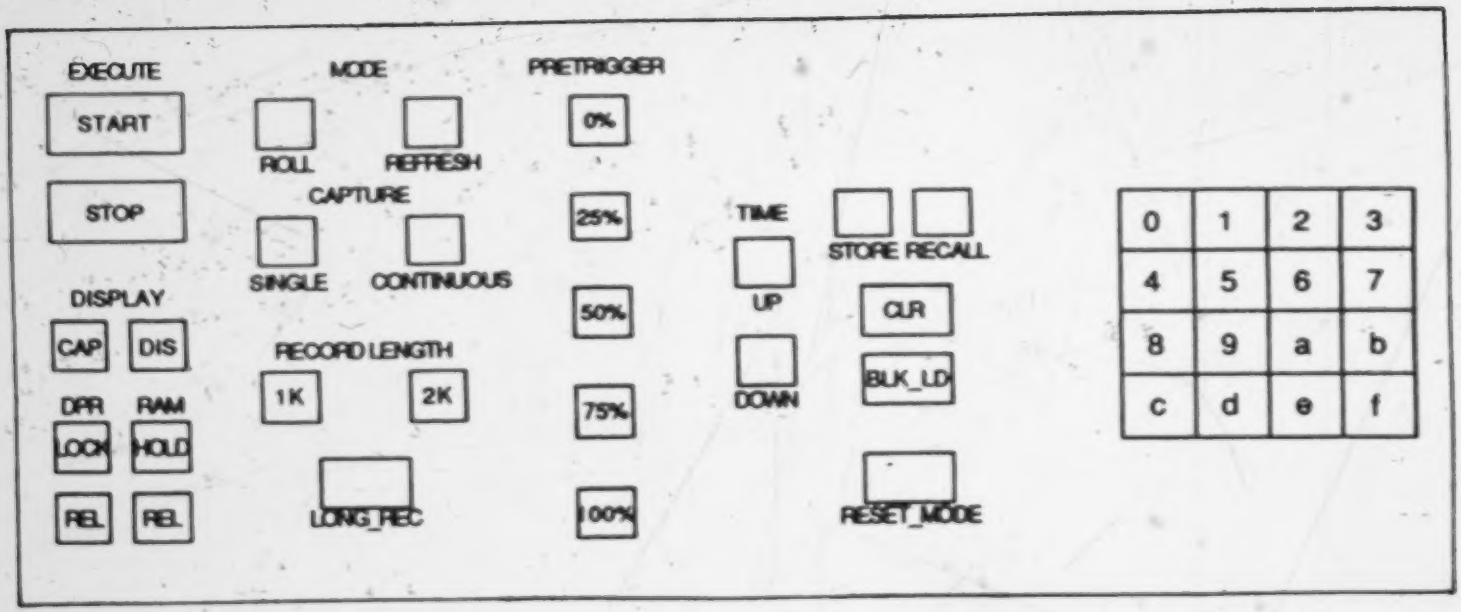


Figure 2.6

The Keypad Format

"SINGLE" CAPTURE button, while continuous capture is chosen with the "CONTINOUS" CAPTURE key. The functioning characteristics of the ADIDAT are dependent on the chosen mode and capture combination, and each has been explained in an earlier section.

With this device, a variable record size is available and is user-selectable by the keypad buttons found under "RECORD LENGTH". The key labelled "1K" allows 1-Kbyte of data to be stored in memory while the "2K" key allows 2-Kbytes of data. Also, the key marked "LONG_REC" enables the user to select a long record consisting of 16-Kbytes.

The method by which data is to be displayed can be controlled by the keys found under "DISPLAY". If the record is to be stored in storage RAM, the user must select the "CAP" (ie. capture) key, however, two unique possiblities exist and are dependent on the MODE of the ADIDAT. If the device is in the "CONTINUOUS CAPTURE" mode, data is stored in the storage RAM as well as the display RAM. This allows the user to see the waveform as it is being stored, but it must be kept in mind that storage will occur only in the specified record length. Therefore, once the memory space assigned for a record has been filled, the input data will begin to over-write the previously stored data. In order to retain data in either of the RAMs without the fear of it being overwritten, a "LOCK" and a "HOLD" key are available for the display RAM and storage RAM respectively. The "REL", or RELEASE key disables the locked or held RAM. For the "SINGLE-SHOT CAPTURE" mode, only the storage RAM is enabled while the dual-port RAM is inoperative. When the "DIS" key of the "DISPLAY" section has been pressed, the ADIDAT is in the transfer mode of operation. This key is used when stored records need to be recalled for display, so the device transfers

data from the storage RAM to the dual-port display memory. When "DIS" has been engaged, the "CAP" key is disengaged, and vice versa.

The keys beneath the "PRETRIGGER" section are labelled according to the amount of data to be stored prior to trigger arrival. Hence, the valid pretrigger settings are 0%, 25%, 50%, 75%, or 100%.

The internally generated timebase can be selected using the "TIME UP" or "TIME DOWN" keys, which effectively step through each of the possible acquisition rates. Pressing the "TIME UP" key once advances the step by one to the next acquisition rate while pressing "TIME DOWN" once causes the acquisition rate to be reduced by one step.

The next group of keys are used in operations involving input values destined for variable-latches and variable-counters. To store variables in a latch, the "STORE" key is used, while clearing a variable is accomplished using the "CLR" key. If a data record is being recalled from storage RAM, the "RECALL" key is used in order to load a counter with the preset value. Finally, the "BLK_LD" button is used to preset the value of a counter with the desired 4-bit memory block value, while the "RESET_MODE" key causes the ADIDAT to go into single capture mode.

The last section of the keypad comprises of a push-button array of numbers from 0 to 9, followed by a to f. These are the hexadecimal equivalents of the decimal numbers 0 to 15. These keys are used to select the memory block where the data record is to be stored.

In order to fully understand the use and operation of the keypad, an example of the required keystrokes for a simple operation is given. In particular,

the example demonstates the single-shot capture of an input waveform in a 1-Kbyte memory block with 25% pretrigger data. Storage data is to be captured in memory block 5 using the roll-mode at a 10 MHz sampling rate. First, the user presses the "CAP" function key followed by the "SINGLE CAPTURE" key to signify that the data is to be stored into storage memory rather than a data transfer from memory. Next, the "ROLL" mode is selected so that pretrigger data can be captured. Thus, the 25% key must be depressed to indicate the amount of data to be stored prior to trigger arrival. Since the record length is only 1-Kbyte, the "1K" key is pressed followed by the "5" key, which signifies the starting point for the data. At this point, the internal variable-clearing line is deselected allowing the new variables to be stored. Following the input of the memory block value, the "BLK_LD" key is pressed in order to load in the selected block value. Next, the "STORE" key is pressed so that all the selected variables may be latched in their corresponding registers. In order to get the desired sampling rate, the "TIME UP" is keyed once so that the sampling frequency is stepped up from 10 KHz to 10 MHz (see Figure 4.10). Finally, after everything has been entered and stored, the "EXECUTE START" button is pressed to initiate the capture sequence. Once the acquisition phase is complete, the L.E.D. indicator light beside block 5 will illuminate, and the user should press the "EXECUTE STOP" key.

2.6.13 MEMORY ELEMENTS:

The ADIDAT circuitry incorporates two different types of memory storage devices in its design. One type of memory element is the standard CMOS static RAM package, which is used as a data storage buffer. The other device is a dual-ported CMOS RAM chip which is used for display memory. The operation of each will be examined in this section.

2.6.12.1 CMOS Static RAM

After an input analog waveform has been processed through an A-D converter, the resulting digital signal may either be displayed immediately, or can be stored in memory for later use. For the storing phase, the incoming data stream of the digitized waveform is split into four separate segments to allow for a RAM multiplexing arrangement. The method by which the storage RAM is partitioned is described in the section entitled "Memory Addressing and RAM Control Circuitry". This scheme implies that the complete record of any input signal would have to be stored in four distinct RAM modules. Thus, the storage memory comprises of eight 4K x 4-bit RAM modules multiplexed in an arrangement to produce four memory blocks, each containing 4-Kbytes, while allowing 8-bit data storage. Multiplexing the resulting four memory blocks enables the use of relatively slow, inexpensive RAM since each memory segment would be accessed only 1/4 of the clock rate.

2.6.12.2 CMOS Dual-Port RAM

The display RAM is a dual-ported memory element, meaning that the actual storage registers can be accessed by two independant sources. Thus, for the 2K x 8-bit dual-port display RAM used with the ADIDAT, there are two ports with separate control, address, and I/O pins that permit independant access for reads or writes to any location in memory [21]. This RAM allows one port to be dedicated to a microprocessor, which cycles through the memory to produce a readable display, while the other port is used by the dedicated logic array to allow data transfers from either the A/D latches or from the storage RAM. Where the data originates from depends on the mode setting of the ADIDAT.

The timing diagrams for both of these RAM devices are shown in Figures 2.7 and 2.8, on the following pages.

THE MATERIAL CONTAINED ON THIS PAGE
HAS BEEN OMITTED DUE TO THE UNAVAILABILITY
OF COPYRIGHT PERMISSION.

This figure taken from "High-Speed CMOS Data Book"
as indicated in reference 21.

Figure 2.7

Timing Diagrams for the Static RAM

THE MATERIAL CONTAINED ON THIS PAGE
HAS BEEN OMITTED DUE TO THE UNAVAILABILITY
OF COPYRIGHT PERMISSION.

This figure taken from "High-Speed CMOS Data Book"
as indicated in reference 21.

Figure 2.8

Timing Diagrams for the Dual-Port RAM

3. Principles of Waveform Storage and Display

An input analog waveform is first processed to produce a digitized version of the input signal. The data obtained from the output of the analog-to-digital converter may either be displayed immediately or be stored for subsequent use.

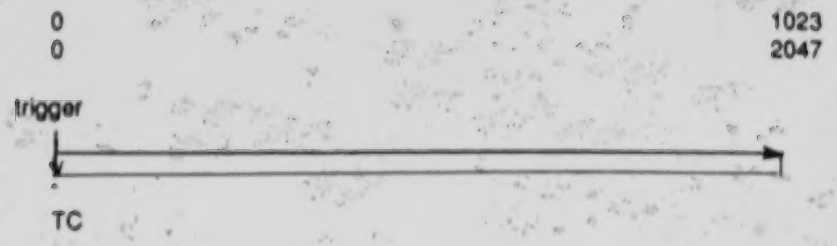
The unique method by which the storing process occurs is under the direct control of two 11-bit counters, which are used in the address generation for the memory elements. The details on counter operation have been fully described in the section entitled "PRETRIGGER CONTROL CIRCUITRY" which follows. The method by which storing occurs depends on the device mode, pretrigger information desired, and when a valid trigger is received. To convey this information, diagrams will be used to show counter operation for each individual case. The diagrams represent the address locations in memory with the top numbers (0 to 1023) corresponding to a 1K record while the bottom numbers refer to locations 0 to 2047 in a 2K record.

There exists a wide variety of methods by which a user may store a digitized waveform, however, all incorporate one basic idea. In order to satisfy pretrigger requirements, address generation continuously cycles from address location 000 (hex) upto the address corresponding to the percent pretrigger desired for a given record. That is, for a 1-Kbyte record and 50% pretrigger, the pretrigger counter cycles from 000 to 200 (hex). This address generation loop continues until a valid trigger is received by the circuit, at which point the pretrigger counter output value is stored, and the counter is disabled. It should be emphasized that the looping of the pretrigger counter does in fact over-write previous, out-dated data. However, the pretrigger cycle will always contain data corresponding to the last 511 data points before the arrival of a trigger. If on the

other hand, the trigger arrives before the first loop has been completed, then the pretrigger requirements will unavoidably not be met. With respect to the earlier example, after the triggering event has occurred, the posttrigger counter will generate the RAM addresses from location 200 (hex) to 400 (hex) thereby not overwriting any previous pretrigger data. In the situation where the trigger arrives prior to the completion of the pretrigger cycle, the post-trigger counter will count from the address where the trigger was encountered, upto 400 (hex).

For the first case, single-shot capturing will be analyzed in which data is stored in the storage RAM so that it may be viewed at a later time. In this situation, only the storage memory is enabled while the display RAM is disabled. If the device is operating in ROLL mode, pretrigger information can be obtained and therefore must be specified. For the REFRESH mode, the pretrigger value automatically defaults to 0%. In the following diagrams, "TC" refers to the terminal count value for the pretrigger counter, while "TRIGGER" refers to the actual time a valid trigger is received by the circuit. In all cases shown below, if the pretrigger counter is halted because of an incoming valid trigger, the last count value will be latched in the block labelled "POST_LATCH". This is done to allow the signal to be correctly displayed, as will be described later in this section. Also, if a trigger does not occur, the pretrigger counter will continue the process of cycling from the zero state to its terminal count value. This procedure for the pretrigger counter can only be halted by the occurrence of a trigger or can be interupted by the user.

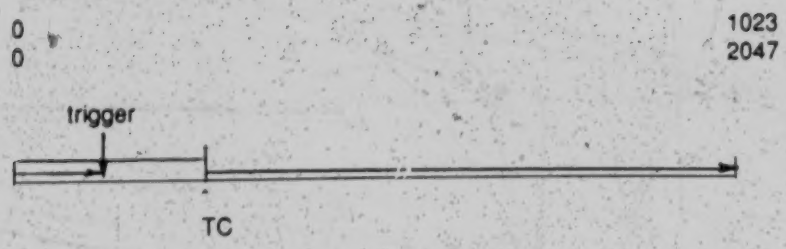
3.1. STORAGE WITH 0% PRETRIGGER



-in this case, the pretrigger counter never starts incrementing, and the posttrigger counter starts only after receipt of a valid trigger. The terminal count for the pretrigger counter is zero.

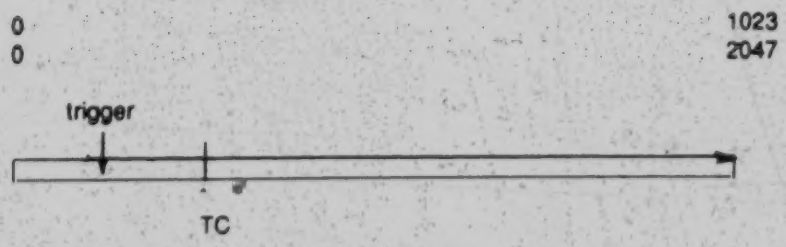
3.2. STORAGE WITH 25% PRETRIGGER

3.2.1. Triggering after pretrigger counter reaches terminal count



-for this situation, the posttrigger and pretrigger counters increment in unison upto the point where the pretrigger terminal count is reached. When this occurs, only the posttrigger counter is disabled at value of 255. Therefore, upon reciept of a trigger, the posttrigger counter is re-started at 255 so as not to overwrite data.

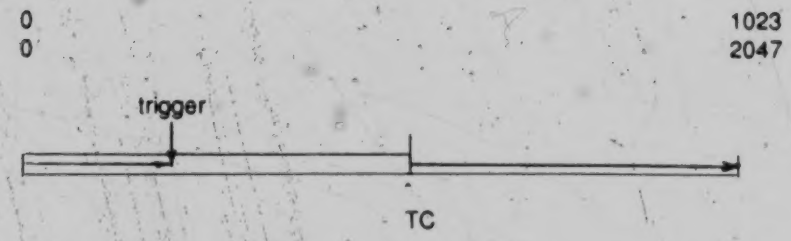
3.2.2. Triggering before pretrigger counter reaches terminal count



-again, both counters are counting together, but if the trigger arrives before the terminal count of the pretrigger counter, then the posttrigger counter continues incrementing from its current value, while the pretrigger counter is disabled.

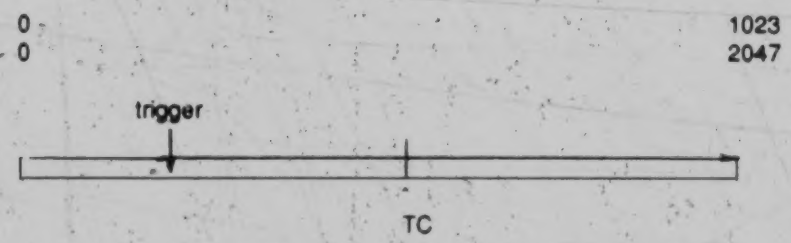
3.3. STORAGE WITH 50% PRETRIGGER

3.3.1. Triggering after pretrigger counter reaches terminal count



-again, the posttrigger and pretrigger counters increment in unison upto the point where the pretrigger terminal value has been reached. When this occurs, only the posttrigger counter is disabled at a value of 511 so that when a trigger is received, it can be re-started from this value. This ensures that pretrigger data is not overwriten.

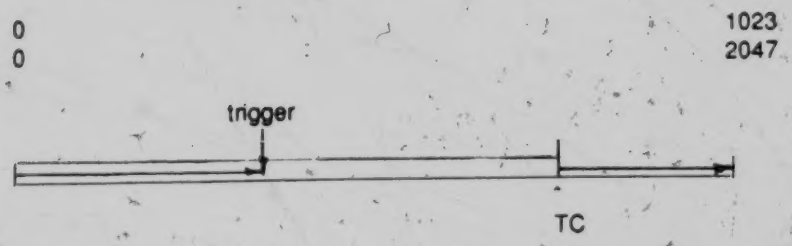
3.3.2. Triggering before pretrigger counter reaches terminal count .



here, both counters are counting together, but if the trigger arrives before the terminal count of the pretrigger counter, then the posttrigger counter increments from its current value while the pretrigger counter is disabled.

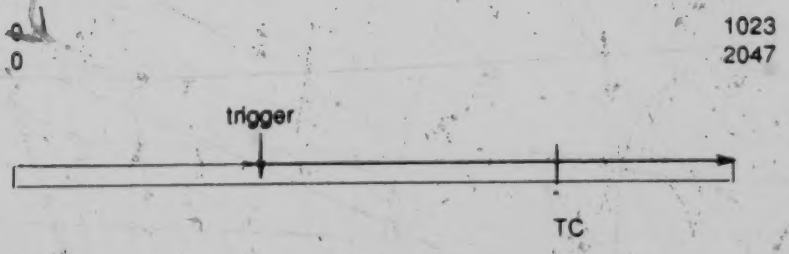
3.4. STORAGE WITH 75% PRETRIGGER

3.4.1. Triggering after pretrigger counter reaches terminal count



-for this situation, the posttrigger and pretrigger counters increment in unison upto the pretrigger counter terminal value. When this occurs, the posttrigger counter is disabled at the value of 767. When a trigger occurs, it is re-started at 767 so that it does not overwrite any data.

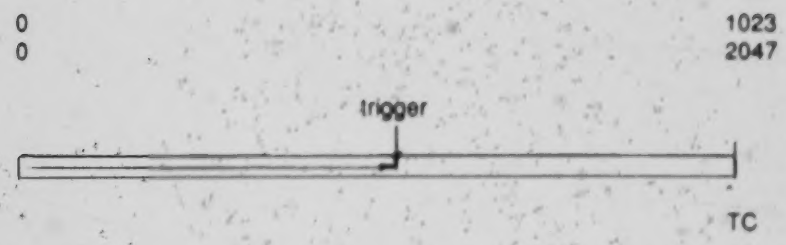
3.4.2. Triggering before pretrigger counter reaches terminal count



-again, both counters are counting together, but if the trigger arrives before the terminal count has been reached, then the posttrigger counter continues incrementing from its current value. During this time, the pretrigger counter has been disabled.

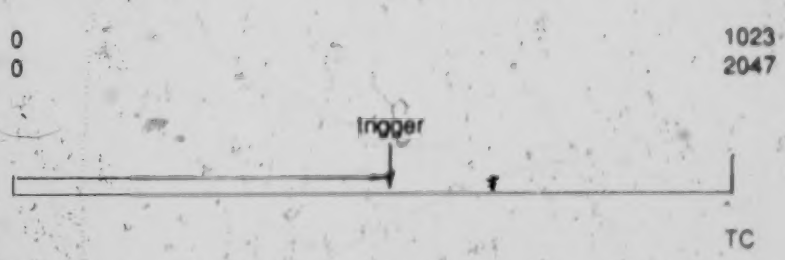
3.5. STORAGE WITH 100% PRETRIGGER

3.5.1. Triggering after pretrigger counter reaches terminal count



-in this situation, the posttrigger counter never begins to increment and all address generation is produced by the pretrigger counter. When a trigger is received, the pretrigger and posttrigger counters are both disabled.

3.5.2. Triggering before pretrigger counter reaches terminal count



-if the trigger point occurs before the pretrigger requirements have been met, then the pretrigger counter is disabled and the remaining amount of memory is left with zero values.

In the second case where an immediate display is desired, the device must be set in the CONTINUOUS-CAPTURE state. For this situation, the pretrigger value defaults to 0% so the only counter in operation is the posttrigger counter. Also, both storage and display RAMs are now enabled, unlike the previous case where only the storage RAM was enabled. If the device is in ROLL mode, the screen displays a rolling waveform like that seen through the window of a chart recorder. It should be noted that the display provides a free running trace and thus, all incoming triggers are ignored. The purpose of enabling both RAMs is so that the data is stored for future use as well as being displayed immediately. If the user wishes to halt the rolling display, the LOCK-screen button is depressed, but this still allows further storage of incoming data. The user may also decide to use the HOLD button, which prevents further data being written into the storage RAM but allows continual updating of the display RAM.

After all required waveforms have been captured in the storage RAM, the next step would be to display the desired signal. During the storing process, the digitized waveform was split into four seperate segments to allow for RAM multiplexing. Thus, a complete record of an input signal would be stored in four distinct RAM modules. In order to display the signal, it would be convenient to re-organize the data so that one complete record is found in only one RAM cell. This is the purpose of the display RAM, where data of only one record is transferred from the storage to the dual-port display RAM. In order to accomplish this, the storing address sequence must be replicated except instead of writing to the RAM locations, a read from the storage memory is performed.

In order to determine which stored waveform is displayed, the user can determine which RAM blocks are filled with data of either 1K or 2K length by examining the led indicators. These lights are controlled by the "RAM_LED" cell

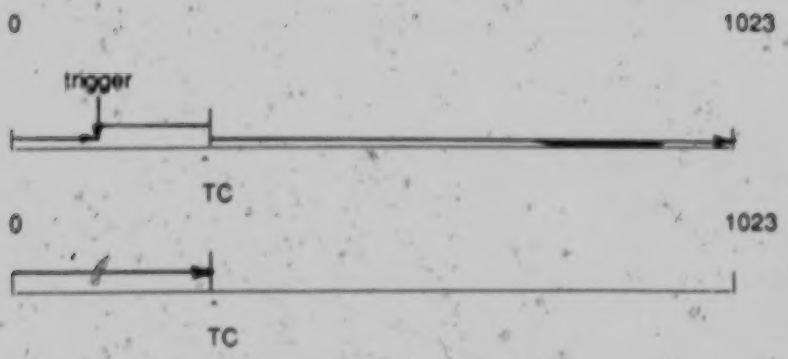
found in "CIRCUIT1", and has been described later. During an acquisition, the user selects a memory block destination indicated by the 4-bit number BLK3 and BLSB (lsb of memory block). This value, along with the bit defining record length (MEM_SEL), are input into "RAM_LED". Also, to indicate that data is being stored, the STORE line is low. If a RAM block is being cleared, the STORE line would be taken high. Upon completion of either data storage or removal, the LATCH line is taken high to either turn off or on the appropriate LED. Thus, with the knowledge of which location the desired waveform is in, the user selects the memory block via the lines BLK3 and BLSB during the data-transfer cycle.

The two 11-bit pretrigger and posttrigger counters are employed once again for address re-generation, except now they are counting in the display mode, where CAP_DIS=1. Both pretrigger and posttrigger counters are presetted with the value of "count", where "count" was the value of the pretrigger counter output when the trigger had occurred during the capture cycle. Recall that this value was stored in the "POST_LATCH" module upon receipt of the trigger. Upon execution, the pretrigger counter increments from its loaded value upto the value which satisfies the pretrigger constraints set by the user. Thus, for a 1K record with 25% pretrigger requested, the counter should increment 255 times. The amount the pretrigger counter has to increment is determined by the memory counter circuitry, suitably named "MEM_CTR". This block is similar to the front-end of the PRETRIGGER circuitry, but is only enabled during a data transfer from the storage RAM to the display memory. The 11-bit counter found in MEM_CTR is similar to the pretrigger counter except that this counter always starts incrementing from the zero state. Thus, for the case of a 25% pretrigger for a 1K record, this counter will count from the zero state to 255. The output of the MEM_CTR circuit is a signal which indicates when the pretrigger requirements have been fulfilled. The output

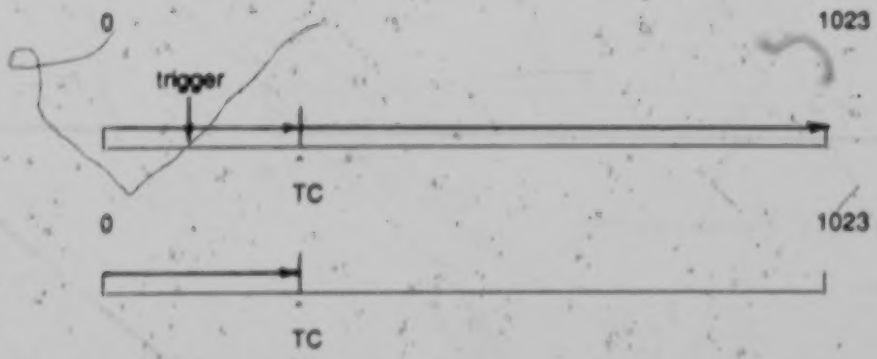
"YD_NOT" is normally high, but goes low when the counter of MEM_CTR has reached its terminal count value. This signal is fed directly into the PRETRIGGER block, and has the same effect as if a trigger were to have occurred. Namely, YD_NOT has the effect of 1) disabling the pretrigger conter, 2) causing the selection of the posttrigger counter, and 3) enabling the posttrigger counter. This has been diagramatically shown on the following page.

3.6. RECALLING WITH 25% PRETRIGGER

3.6.1. Triggering after pretrigger counter reaches terminal count



3.6.2. Triggering before pretrigger counter reaches terminal count



-for both of the cases, the pretrigger and posttrigger counters are loaded with the value stored in POST_LATCH, and they begin incrementing from that point. However, in the first situation, POST_LATCH holds the value of the counter when the trigger occurred, while in the second case, the latch holds the value of zero.

-for the situation where a trigger occurs before the terminal count, during display, the pretrigger counter must count all the way to its' terminal count as dictated by the MEM_CTR circuitry. This causes no problems as normally, the postrigger counter would have counted after the trigger, but since TC had not been reached, both counters are incrementing in unison.

Once the counters reach their final values, the ACQCOMP line goes active low, indicating that the data-transfer cycle is complete. Thus, with a complete record now contained within one memory element, the microprocessor can cycle through the addresses at a rate such that the display will appear constant. With the advantage of a dual-port RAM display memory, one side is dedicated to data input, whether it is directly from the A/D converter latches, or from the storage RAM. The other port is restricted to microprocessor access and all data lines are output either to the processor or a CRT controller.

4. Major Circuit Blocks of the ADIDAT

This chapter deals with the detailed description of all major circuit blocks designed for the ADIDAT dedicated IC [19]. Reference should be made with the glossary, found at the beginning of this thesis, to assist in determining the role of the signal or module being discussed.

4.1 Pretrigger Control Circuitry

It has been mentioned earlier that all memory address generation is controlled through the use of 11-bit programmable counters. The pretrigger circuitry is a crucial element in this design as it provides the necessary logic to properly control these counters.

The pretrigger control circuit is shown in the block labelled "PRETRIGGER", consisting of two 11-bit counters, multiplexers, and associated logic circuitry. The detailed block diagram is shown in Figure 4.1. The primary functions of this circuit can be summarized by the following:

1/ to ensure that enough data has been acquired by the storage memory to satisfy the user-defined pretrigger requirements.

2/ after a valid trigger has been received, the required amount of posttrigger data must be written into memory to generate either a 1K or 2K record length.

3/ the circuitry must be able to terminate the acquisition phase so as to prevent further data being written into the storage RAM.

4/ allow data transfer to the display RAM using the same logic circuitry used during the acquisition sequence.

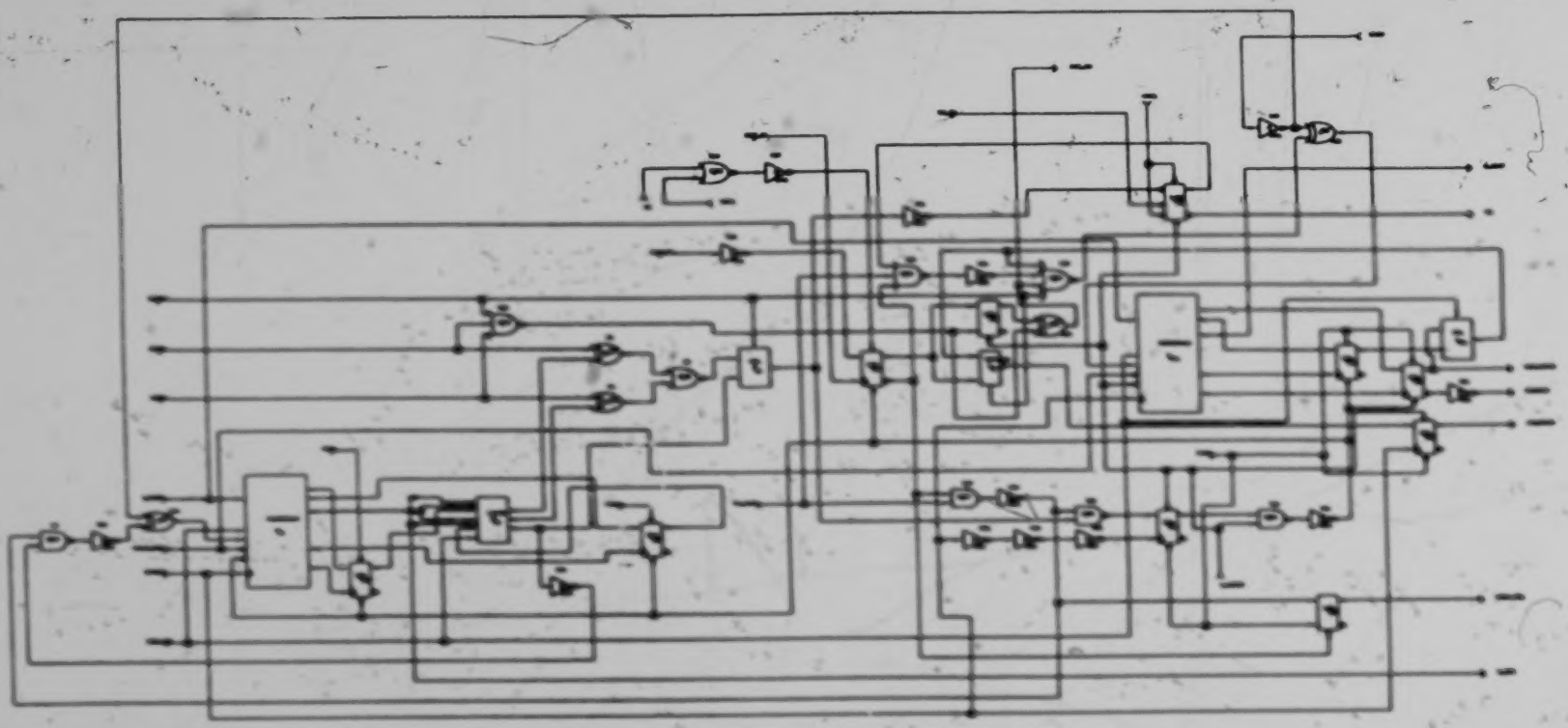


Figure 4.1

The PRETRIGGER Module

The two 11-bit counters form the basis of the control circuit, where U1 is called the pretrigger counter and U14 is the posttrigger counter. Both counters increment on the rising edge of the internal clock, with the former counting before trigger arrival, and the latter incrementing after trigger arrival. The logic must therefore select which counter determines memory addressing and what the count length for each will be. To ensure proper re-display of the waveform, circuit performance should be identical for either data capture or data transfer phases. However, for data transfer sequences, all externally generated incoming triggers must be ignored. The circuitry within the MEM_CTR module, however, produces a signal (YD_NOT) which simulates a triggering pulse. The signal is required since [illegible]er control is highly dependant on a triggering signal. This signal is fed into the PRETRIGGER module in order to produce identical circuit performance for both data acquisitions and transfers. Circuit performance will now be discussed in general, after which circuit operation will be examined in detail.

Referring to the PRETRIGGER block in Figure 4.1, it is evident that most of the logic circuitry is used for the enabling and clearing of the pretrigger and posttrigger counters. Since these counters have the unique ability to vary their count lengths for 1K or 2K record lengths, multiplexers are used to select the proper counter output bits. The multiplexer select line is controlled by the same variable which selects record length, namely "MEM_SEL". In this way, either a 1-Kbyte or 2-Kbyte record length can be chosen. To properly understand the operation of this circuit, reference should first be made to the section on "PRINCIPLES OF WAVEFORM STORAGE AND DISPLAY" found in Chapter 3. Initially, both counters are reset to zero and are enabled so that they both may count together in unison. This condition can only be altered by the receipt of a valid trigger or the re-starting of the pretrigger counter at zero after it has reached

its terminal count value. The terminal count value to which the pretrigger counter increments depends on the pretrigger setting. The pretrigger logic comprises of exclusive-ORs U3 and U4, NOR gates U5 and U32, and a data selecter U7. This has been shown in Figure 4.2. The pretrigger values are fed into this circuit through the variables PRE25, PRE50, and PRE100. In order to obtain proper pretrigger information, the 2 most-significant bits plus the carry-out of the pretrigger counter need to be accessed. This is independent of the record length selected, so if "MEM_SEL" is 0, a 1K record is chosen and Q8, Q9, and SHORT_CO (carry-out for a 1023 count) are accessed by the control circuit. The Q8 output of the counter will therefore go high after a total of 255 clocks, the Q9 output goes high after 511 clocks, and the SHORT_CO after 1023 clock cycles. Similarily, if "MEM_SEL" is 1, a 2K record is selected so Q9, Q10, and CO (carry-out for a 2047 count) must be accessible by the logic circuitry. The Q9 output goes high after 511 clock pulses, Q10 after 1023 clocks, and CO after 2047 clock cycles. From this, it is easy to see the function of the data selector, U2. For a 1-Kbyte record, inputs A0, A1, and A2 corresponding to Q8, Q9, and SHORT_CO are seen at the output, while for 2-Kbyte storage, the output follows inputs B0, B1, and B2 which are tied to Q9, Q10, and CO respectively.

In the case of the posttrigger counter, U14, only CO or SHORT_CO need be accessed by the count termination control circuitry. These signals merely indicate the completion of an acquisition phase through the "ACQCOMP" (active low) output of U18. Again, the appropriate signal is selected by using data selecter, U25, and controlled by the "MEM_SEL" line.

An acquisition is initiated by a momentary low logic level occurring at the R_RESET input, which causes a reset of both counters and all pertinant flip-flops. The resetting process causes "ACQCOMP" to be set high, thereby indicating the

start of an acquisition. Circuit behaviour now depends on the pretrigger value selected, which is determined by the PRE25, PRE50, and PRE100 signals provided by the user. It should be mentioned again that this circuit is employed both during an acquisition phase as well as a re-display phase. During data capture, the input trigger is received via the D-type latch, U8. This has been defined as the trigger latch in subsequent discussions. Also during an acquisition, the input "YD_NOT" is always high, as it arises from circuitry which is enabled only during data display (refer to the MEM_CTR block). During the re-display phase, all triggers which may occur are ignored since the trigger latch is effectively disabled. The value for "YD_NOT" is generated through the enabled MEM_CTR block, and acts like an internally generated trigger. This scheme allows either the trigger (in a capture sequence) or "YD_NOT" (in a data transfer phase) to determine which counter outputs will be treated as the memory address, depending on whether the device is capturing or displaying data. Counter output selection is performed in the SELECTOR module, although the variable "CNTR_SEL" which performs the selection is generated within PRETRIGGER.

In the following discussion, it is assumed that the "MEM_SEL" signal is low which implies a 1K record length. As a result, multiplexer U2 will select counter outputs Q8, Q9, and SHORT_CO, while data selector, U25 will select the SHORT_CO output from the posttrigger counter.

4.1.1 0% PRETRIGGER

For no pretrigger requirements, PRE25, PRE50, and PRE100 are all set low, thus the PRE25 and PRE50 inputs to the exclusive-OR gates U3 and U4 are low. Since the counter has been initially reset, its corresponding outputs are all cleared so that the other remaining inputs to the XOR pair are also low. This

produces a high value at the output of the NOR gate, U5. The 2-bit selector ("SEL") which follows is controlled by the PRE100 line, therefore the output of U7 would follow the value on input line "A", which is a logic 1. This can be seen from Figure 4.2 shown on the following page. Now, since both the trigger latch, U8 and the MEM_CTR circuitry were reset initially, both inputs to U10 would be high (YD_NOT=1). This takes the D-input of U13 low because of the action of NAND gate U12 and the high output form U7. On the next negative transition of the internal clock, the Q-output of the reset latch will follow the D-input, which forces an internal reset of U1, U6, U8, U17, U18, and U29, as shown in Figure 4.3.

The output of U20 "enables" the pretrigger counter through U0, U11, and U44, however, this is meaningless since the counter is being continuously reset by the reset latch, U13. This situation causes the pretrigger counter to remain at 000 (hex), and maintains this state until a valid trigger occurs. The XOR gate U44 is used only during system initialization so that the pretrigger counter is not enabled during this time. The pretrigger enabling logic circuit has been shown in Figure 4.4. When a trigger pulse is received by the circuitry, trigger latch U8 is clocked high causing the Q' output of the same flip flop to go low. This also causes the counter select line ("CNTR_SEL"), determined by U16 (which was initially high) to be cleared, thereby selecting the posttrigger counter in the SELECTOR circuitry. Recall that "CNTR_SEL" was initially high so that the pretrigger counter could be selected. Also, the output of U20 is now low causing the output of nand gate U12 to go high, therefore no internal resetting can take place through the reset latch, U13. However, the enable input to the pretrigger counter is now low (previosly high) as a result of U0, U11, and U44, so this counter is disabled. The enabling circitry for the posttrigger counter will now be examined, and is shown in Figure 4.5. The inputs to the 3-input NAND gate, U24, were all high before the

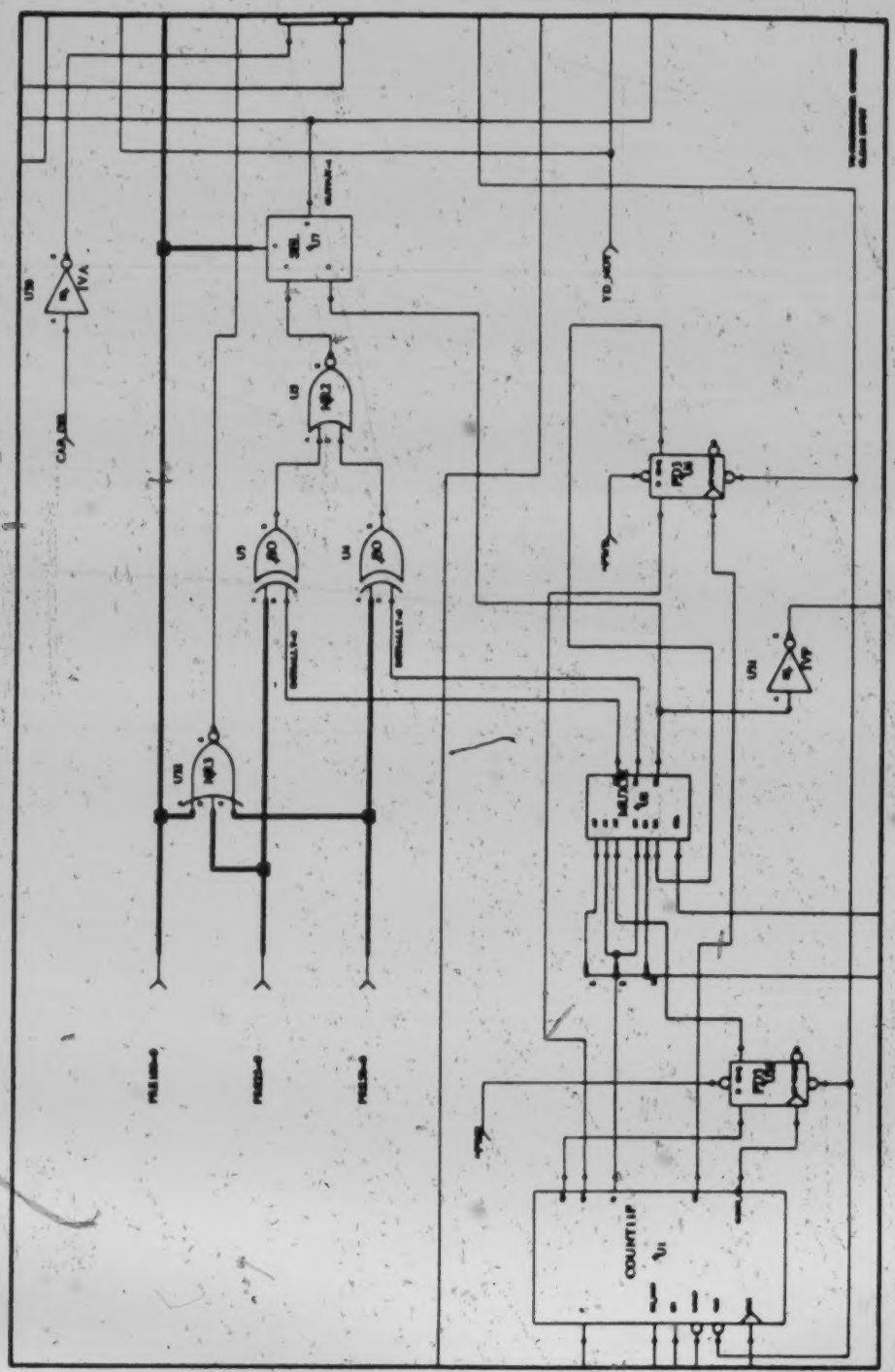


Figure 4.2

Exclusive-Or Circuitry for 0% Pretrigger

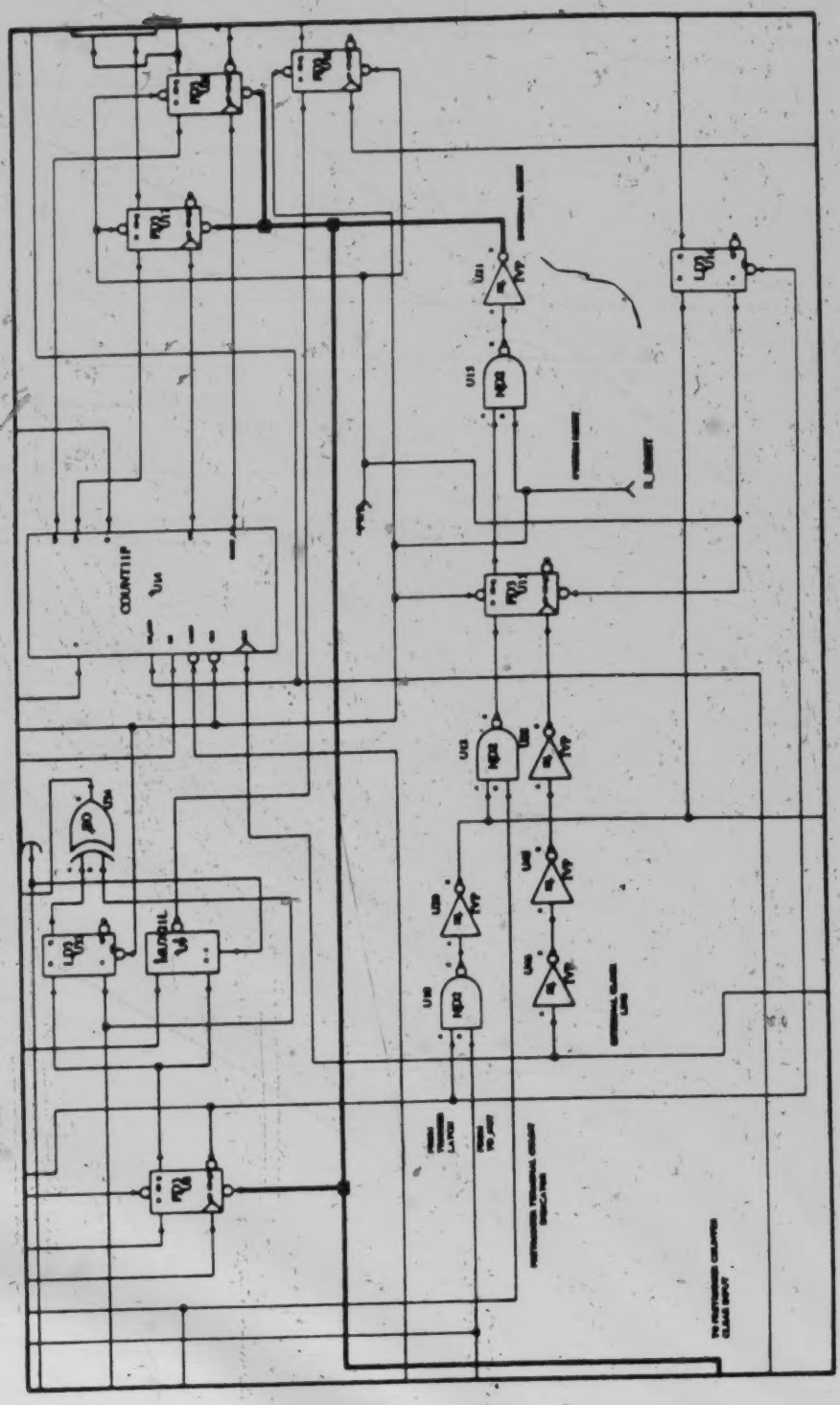


Figure 4.3
Internal Reset Circuitry

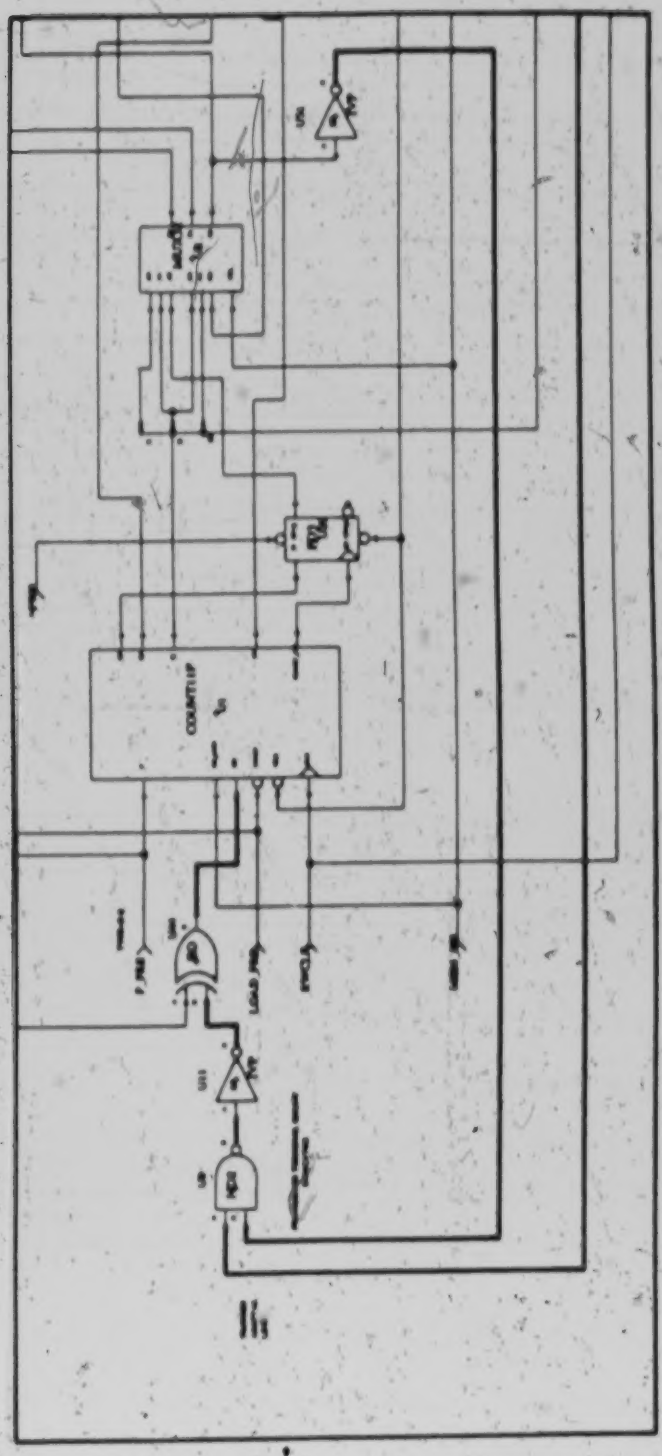


Figure 4.4

Pretrigger Enabling Circuitry

receipt of a valid trigger, and since all inputs, other than the output of U26 to the 4-input NOR gate are low, the posttrigger counter was disabled. But since Q' of the trigger latch is now low (U8), the output of U26 also goes low. This causes the output of the 4-input NOR gate to go high since all inputs to U27 are low, thereby enabling the previously disabled posttrigger counter. The XOR gate U42 is used only during the initialization of the ADIDAT. The counter begins counting from the all zeros state at the selected clock rate. When the counter reaches a count of 1023 the SHORT_CO will go high in turn causing the output of the selector U25 to go high. The inverting multiplexer U9 selects the "A" input, thus its output is low because of U25. The D-input of the ACQCOMP flip flop U18 is thus low, so on the next clock cycle the Q-output will be low. The "ACQCOMP" signal is now low indicating the termination of the acquisition phase. The high value output from selector U25 also sends the output of the 4-input NOR gate low, thereby disabling the posttrigger counter.

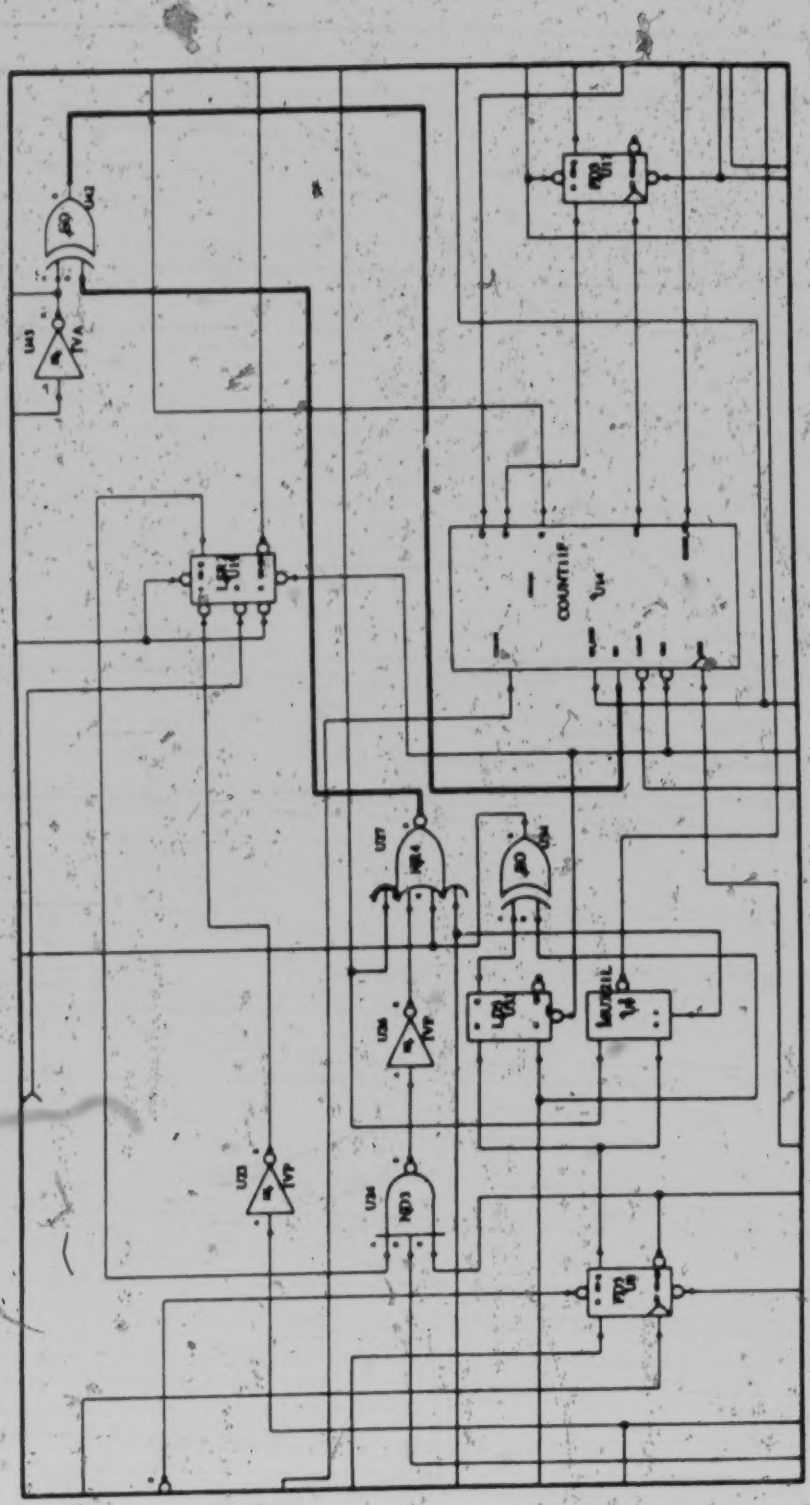


Figure 4.5 Posttrigger Enabling Circuitry

4.1.2 25% PRETRIGGER

For this case, the signal PRE25 is high while the other pretrigger inputs are all low. Since the pretrigger counter was initially reset, both inputs to U4 are zero resulting in a low output, while the output for U3 will be high. Thus the output of NOR gate U5 will remain low until both inputs to exclusive-OR U3 are high. Again, the selector U7 chooses the "A" input which is presently low thereby taking the output of U12 high. On the next negative clock pulse, the Q-output of reset latch U13 is brought high causing no internal resetting. Since the Q' output of the trigger latch and "YD_NOT" are both high, the output of U20 is high. This, along with the initial resetting of flip flops U6 and U29, causes the output of inverter U31 to be high thereby enabling the pretrigger counter. The clocking causes this counter to increment from the all zeros state at a rate corresponding to the frequency of the internal clock signal. When the count reaches a value of 255, Q8 of the pretrigger counter will go high, causing the output of XOR gate U3 to go high,and hence causing the output of NOR gate U5 to go high. Prior to this occurrence, the posttrigger counter was counting in unison with the pretrigger counter since all inputs to the 4-input NOR gate, U27 were low. This resulted in a high value reaching the posttrigger counter enable input, which is active high. However, with selector U7 going high, the combination of U25, U19, and U24 causes U27 to go low, thereby disabling the posttrigger counter until the receipt of a valid trigger. Since the posttrigger counter was disabled and not cleared, its present count value is held for later use.

Having reached the terminal count for the pretrigger counter, the internal resetting process is repeated and the counter starts incrementing from the all zeros state once again. The posttrigger counter remains in its disabled state.

This process continues until triggering occurs, which causes a high logic level to be clocked out of the trigger latch. At this point, the output Q' of U8 will be low causing "CNTR_SEL" to be low and also disabling the pretrigger counter. The output of the 4-input NOR gate U27 will now be high once again "enabling" the posttrigger counter. The count will commence at 255 and continue until it reaches 1023, at which point the "SHORT_CO" of the posttrigger counter will go high. This transition is detected by the inverting multiplexer U9, and thus "ACQCOMP" will go low after the next clock pulse. Again, this will indicate the termination of the capture phase, and the posttrigger counter will be disabled.

4.1.3 50% and 75% PRETRIGGER

The circuit performance for these cases is identical to that described above except for the pretrigger input values. For a 50% pretrigger, PRE50 is high while all other inputs are low, whereas for a 75% pretrigger, the input lines PRE25 and PRE50 are taken high keeping the PRE100 input low. Therefore, the output of U7 will go high when Q9 goes high for the case of 50% pretrigger. Also, the posttrigger counter will be disabled at a count value of 511 and re-started upon receipt of a trigger. For a 75% pretrigger requirement, when both Q8 and Q9 go high, the output of U7 will go high signifying the terminal count for the pretrigger counter. Here, 767 will be the count value for the posttrigger counter, where it is disabled until triggering occurs. These are the only basic differences with respect to circuit operation in comparison with the 25% pretrigger option.

4.1.4 100% PRETRIGGER

For this case, PRE100 is high while all other pretrigger inputs are held low. The major difference between this situation and those described previously is that the selectors U7 and U25 choose the "B" input and that the output of the 4-input

NOR gate U27 is always low. This means that the posttrigger counter is never enabled, which is reasonable to expect since we only want to see and record everything before the occurrence of a valid trigger. Initially, the output of U7 is low since everything has been reset, thus the pretrigger counter increments at the clocking rate. When the count reaches its' terminal value of 1023, the "SHORT_CO" of the pretrigger counter goes high, which causes the output of U7 to go high. On the next clock pulse, the Q-output of the reset latch, U13, will follow the low value appearing at the D-input and an internal reset results. The pretrigger counter resumes counting from the all zero state until a trigger occurs, at which point a high value is clocked out of the trigger latch. Since multiplexer U9 is selecting and inverting the "B" input, which is the output of the trigger latch, the signal "ACQCOMP" will therefore be taken low on the next clock pulse. This indicates the completion of an acquisition sequence.

4.1.5 INCREASED RECORD LENGTHS

This discussion on pretriggering can also be extended to the case where "MEM_SEL"=1, indicating a 2-Kbyte capture record. In this situation, multiplexer U2 will select pretrigger counter outputs Q9, Q10, and CO while data selector U25 selects the CO carry line from the posttrigger counter. Also, for both counters, the 11th bit is enabled allowing a count length of 2047.

4.1.6 OTHER CIRCUIT COMPONENTS

Some logic components in the PRETRIGGER circuit which were not fully discussed will now be described here in detail. Latches U6, U17, U24, and U25 were used to hold the values from the carry lines resulting from counter operation. This is necessary for circuit operation since the carry lines go high for only one clock cycle, then return back to a logic 0.

The effective OR gate formed by NOR gate U36 and inverter U37 was used to disable the trigger latch only when the continuous capture method during ROLL mode had been requested. Disabling here means the trigger latch output was always set to a logic 1. This is the only time during the data acquisition phase when the trigger is disabled. During data display, any incoming triggers should be disregarded by the logic, therefore the D-input to the trigger latch is kept low. This is accomplished by the "CAP_DIS" line feeding the inverter U30.

The combination of gates U24 and U26 (effective AND gate) along with U27 (4-input NOR gate) forms the circuitry which enables the posttrigger counter at the correct times. Recall that this counter becomes active during the incrementing period of the pretrigger counter. This occurs until the pretrigger counter reaches its terminal count value prior to trigger arrival. The posttrigger counter is disabled until a valid trigger reaches the circuitry. The enabled posttrigger counter increments to its terminal count value of either 1023 or 2047, at which time the counter is disabled. For the condition of 0% pretrigger, latch U33, XOR gate U34, and NOR gate U32 are required for proper posttrigger counter enabling. Recall that in this state, the pretrigger counter never increments from its zero state and the posttrigger counter only increments after a triggering event has occurred. Also, during this condition, "YD_NOT"=0, thus the output of inverter U26 would be zero. This would enable the posttrigger counter before a valid trigger had occurred; an undesireable situation. With the addition of U32, U33 and U34, this situation is avoided by enabling the posttrigger counter only after the receipt of a valid trigger.

4.2 The Memory Counter Block

The purpose of the MEM_CTR module is to monitor the pretrigger count value during a data transfer sequence. Since the pretrigger counter does not necessarily start incrementing from an all zeros state during data transfer, a memory counter which measures pretrigger record length must be utilized. It is used to determine the point in memory where the pretrigger record terminates. The MEM_CTR circuit module is therefore active only during data transfers, that is, when the CAP_DIS line is high; otherwise it is disabled. To properly understand the function of this circuit block, reference should be made to Chapter 3.

The MEM_CTR logic is similar to the front-end circuitry found within the PRETRIGGER block. An 11-bit programmable counter, U0, is used to increment from 000 to the selected pretrigger count value. This memory counter works in conjunction with XOR circuitry needed to determine the terminal count for a particular pretrigger setting. The logic comprises of two 2-input exclusive-OR gates, U3 and U4, along with NOR gate U5. The lines PRE25 and PRE50 each feed one input of the exclusive-OR gates, while PRE100 is used as the select line for data multiplexer U6. As was done in the pretrigger circuit, a data multiplexer was used to determine which counter outputs feed the other exclusive-OR gate inputs and the output inverting multiplexer. This was done to facilitate either a 1-Kbyte or 2-Kbyte record length, with the appropriate bits being chosen through the data selector with the control line determined by "MEM_SEL". The single output of this circuit, "YD_NOT", is used in the PRETRIGGER control circuitry as shown in Figure 4.6. The value of YD_NOT arises from the output of the inverting multiplexer, U6. Thus, to indicate a terminal count situation for the memory counter, the value of "YD_NOT" will be low otherwise it remains high. Also,

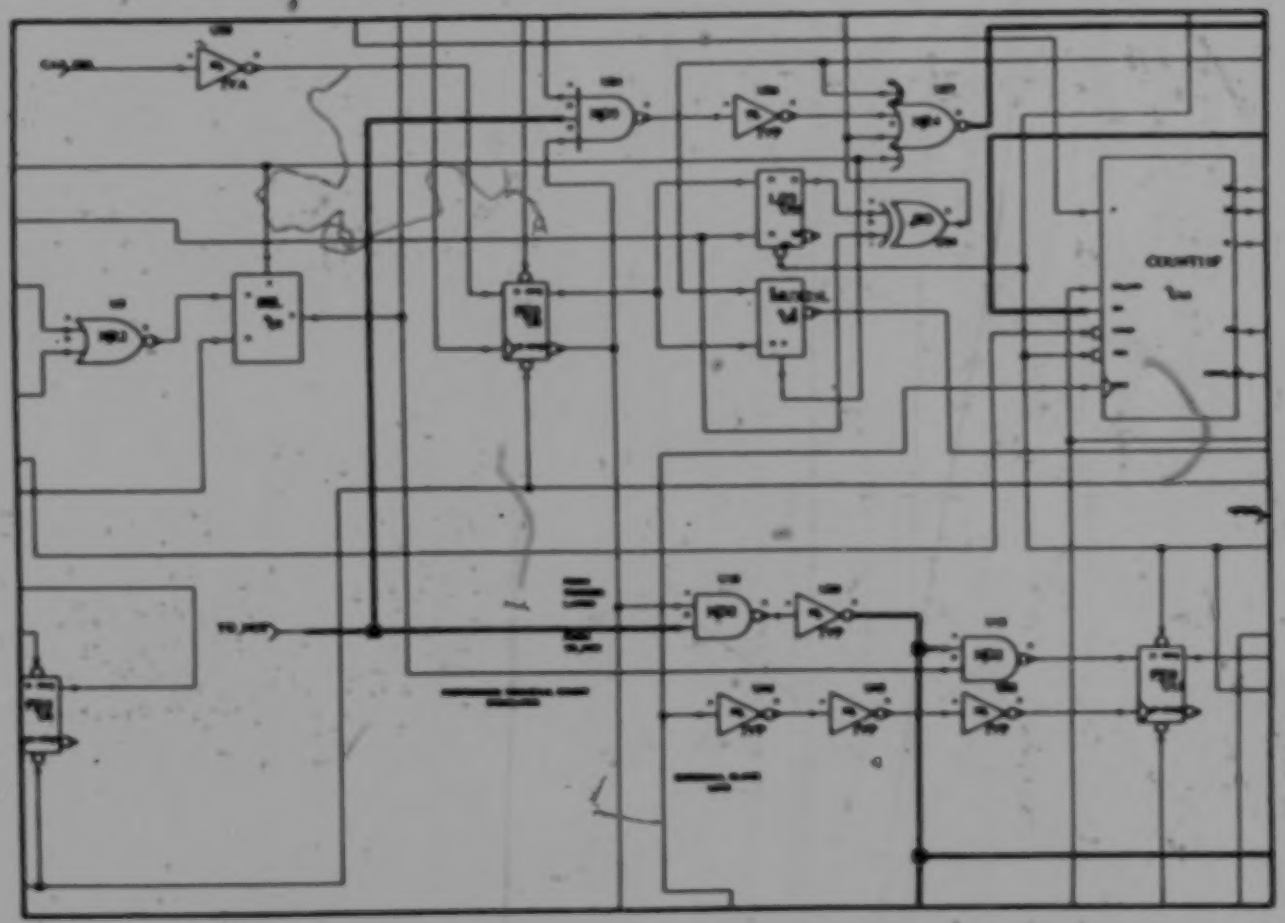


Figure 4.6

YD_NOT in The Pretrigger Module

during data acquisitions when MEM_CTR is not enabled, "YD_NOT" is always at a high logic level.

The method by which this module operates will now be examined. Recall that during data acquisitions the pretrigger counter increments to its terminal value, then resets to an all zero state provided no incoming triggers have occured. This process continues until triggering, at which point the pretrigger counter is halted and the posttrigger counter is enabled. However, the posttrigger counter begins incrementing from the pretrigger terminal count value so that the generated addresses will not result in the overwriting of previously stored pretrigger information. The count continues to either 1023 or 2047 depending on the record length selected by the user. As an example, for a 1-Kbyte record with 50% pretrigger, the pretrigger counter generates addresses from 0 to its terminal decimal value of 511. If no triggers have arrived before reaching 511, the counter loops back to 0 because of the action of the internal reset circuitry. The count continues until a valid trigger is received. If it is assumed that this occurs at address 200, then this last pretrigger count value is stored in "POST_LATCH" (refer to the section entitled "LATCHES") and the posttrigger counter is enabled and begins counting from 512.

When a stored data record needs to be transferred into display RAM, the user sets the "CAP_DIS" line high thereby enabling the MEM_CTR circuit block. The pretrigger counter is loaded with the value "count" corresponding to the address where the pretrigger counter was halted upon trigger arrival. The next point (ie. "count"+1) is the value at which the pretrigger record begins, and continues incrementing upto its terminal count, which has been defined to be "count". To determine when the pretrigger counter is to be stopped, it is

necessary to monitor the memory count value for "n" incremental steps, where "n" refers to the selected pretrigger length. The monitoring task is accomplished by the MEM_CTR circuit, where the 11-bit counter contained within it counts from a cleared initial state to the pretrigger length. Upon reaching the terminal count condition for the memory counter, the output "YD_NOT" is taken low. This action causes the same result that would happen if a trigger was received by the PRETRIGGER circuit. Thus, "YD_NOT" can be regarded as an internally generated trigger. In particular, receipt of a low logic level on "YD_NOT" causes the posttrigger counter to be selected for address generation and eliminates the occurence of an internal reset through U13 in the PRETRIGGER module. Also, this action allows the posttrigger counter to be enabled while disabling the pretrigger counter. The value of "YD_NOT" remains low until data transfer is complete and the device is externally reset by the initiation of another data handling sequence. With reference to the earlier example for a data transfer phase, the pretrigger counter is first loaded with count=201, and then allowed to count upto its terminal count value of 511. Meanwhile, the memory counter is incrementing at the same clock rate from an all zeros state upto 511. When the pretrigger counter has reached address 511, the memory counter has only attained a value of 310. Thus, since the internal trigger line "YD_NOT" is still high, the PRETRIGGER circuitry initiates an internal reset sequence. This causes the pretrigger counter to re-start from the zero state, while the memory counter increments from 311. When the memory counter reaches 511, "YD_NOT" goes low thereby disabling the pretrigger counter, which would have reached address 200. This value corresponds to the end of a pretrigger record. The posttrigger counter is enabled and starts incrementing from 512 and continues to 1023.

The situations described above assume that trigger arrival occurs after the pretrigger counter has reached its terminal count value and has reset back to 0. However, if a valid trigger arrives before the terminal count value, then a different situation arises. Recall that the address where the pretrigger counter was halted upon trigger arrival is stored in the module "POST_LATCH". But latching only happens if the pretrigger terminal count was reached, otherwise the value stored within the latch remains at 0 for that particular record. In this early trigger arrival situation, data transfer is accomplished by loading the pretrigger counter with 0 and letting it count in unison with the memory counter. Incrementing will occur until the terminal count of both counters is reached, at which time the posttrigger counter will assume address generation. It is important to note here that in this situation, the pretrigger counter is generating addresses for both pretrigger data and a partial amount of posttrigger information. This is not a problem and can be readily seen from the diagrams contained within the "PRINCIPLES OF WAVEFORM STORAGE AND DISPLAY" section.

4.3 Memory Addressing and RAM Control Circuitry

After an input analog waveform has been processed through a high-speed analog-to-digital converter, the resulting digital signal may be displayed immediately, or can be stored in memory for later use. With digital storage oscilloscope operation, the latter situation occurs more frequently than the former. Since the digital data stream which results after A-D conversion can be as high as 20 MHz, a high-speed RAM chip must be used. However, in order to reduce the costs incurred with fast RAM packages, an alternative method of data storage can be employed [4].

By using an n-RAM multplexing scheme, where "n" refers to the number of RAM cells used, data is stored into each of the memory modules at a rate corresponding to 1/n of the internal clock speed. This facilitates the use of the more conventional RAM packages which tend to be slower, but are therefore significantly cheaper. Using this technique of data storage, logic must be designed so that addressing of memory occurs in an orderly and systematic manner. Such a task involves addressing each memory element for a specified amount of time in order to allow data to be properly stored in that RAM cell. This suggests that address and data latching may be required to produce the desired effects. Also, in order to enable the memory elements sequentially such that only one RAM cell is active for one given address, an n-bit shift register can be used. Therefore, only 1 of the "n" memory cells will be enabled for one clock cycle, but all memory components will be accessed by the same address location after "n" clock pulses. After the shift-register has enabled each RAM module, the next sequential address is generated and the enabling process is re-initiated. With n-RAM multiplexing, a digitized record would be seperated into "n" distinct sections. Therefore, both the RAM control circuit and PRETRIGGER logic block must be able to reproduce the original signal from the divided records using proper re-construction techniques. This can be easily accomplished since the addressing counters of the PRETRIGGER circuit produce the same count sequence during a data transfer phase as during a data acquisition phase. Thus, the order of memory location addressing is maintained because of identical count incrementing and the resulting circuit operation. To convert the counter outputs into the correct multiplexed RAM addresses, control circuitry is required, and will now be examined.

The majority of the RAM control circuitry is found within the block labelled "SELECTOR", although some signals do originate from other logic modules. The logic contained within these blocks is used to control both the storage RAM and the dual-port display RAM, depending on the user-selectable options. The user can chose either single-shot storage, where data is latched and then transferred to the storage RAM, or use the ADIDAT in a continuous capture mode, in which case data is transferred to both storage and display RAM.

The storage memory comprises of eight 4K x 4-bit RAM modules multiplexed in a scheme to allow 16-Kbytes of 8-bit data to be stored. Multiplexing the resulting four memory modules allows the use of relatively slow RAM since each memory block would be accessed only 1/4 of the clock time. The display RAM is a dual-ported memory element, meaning that the actual storage registers can be accessed by two independant sources. Thus, for a 2K x 8-bit dual-port RAM, there are two sets of the following: 8-bit data lines, 11-bit address lines, and various control lines such as chip enables, write enables and output enables. This RAM allows one port to be dedicated to a microprocessor, which cycles through the memory to produce a readable display, while the other port is used by the dedicated logic array to allow data transfers from either the A/D latches or from the storage RAM. Where the data originates from depends on the mode setting of the ADIDAT.

Fast address generation for both RAMs is accomplished by the use of two 11-bit counters originating from the PRETRIGGER circuit block, and can operate at frequencies up to 20 MHz. These addresses are latched by the RAM_ADD circuitry of U32, which consists of an independently clocked 12-bit latch. The reason for using 12-bit latches when the count is only 11-bits will be discussed

later. Similarily, the A/D latches are formed by the DATA_REG block, U0, consisting of four 8-bit tri-statable data registers. The latching process is governed by the same RAM-enabling shift-register outputs, and is used by DATA_REG as input clocks. The outputs of the 8-bit analog-to-digital converter, representing the instantaneous amplitude of the input signal in digital form, forms the data to be latched. The data is input to the DATA_REG through line "DATAIN". Since the A/D converter operates at a constant rate of 20 MHz, data is continuously generated at 50 ns. intervals. However, both the RAM_ADD block and DATA_REG module latch only on the rising edge of the internally generated input clock(s). Thus, the data register stores only 1 in "x" data samples depending upon the selected acquisition rate. These four latches subsequently drive the four seperate data lines of the storage RAMs [4]. However, when the display RAM must write data to memory, it comes directly from the A/D converter output. These data lines need to be tri-statable so that bus contentions with data lines from the storage RAM can be avoided. For more information please refer to the memory block diagram which follows this section.

The method by which the pretrigger and posttrigger counters address RAM memory locations will now be examined. Reference should be made to the logic schematics contained within the "SELECTOR" module. The 11-bit counter outputs are fed into an 11 x 2:1 multiplexer, U1, where the output is selected by the "CNTR_SEL" line originating from the PRETRIGGER block. The outputs of ADD_MUX multiplexer U2, which follow U1, depends on the record length selected by the user. Recall that for a 1-Kbyte record, the counter would normally only count a 10-bit number and the 11th bit, in this case, would always be zero. Thus, a slight variation of the 11 x 2:1 multiplexer produced the ADD_MUX module, which was used to determine if the MSB of the counter was significant.

This data selector chose between two 10-bit numbers and two 1-bit numbers. The most significant counter bit was one input and the other input was the LSB of the selected memory block. To maintain timing, the remaining 10-bits were also muxed with themselves, so that the result would appear at the output at the same time. Selection between the MSB of the counter and the LSB of the memory block was determined using the "MEM_SEL" line. If "MEM_SEL" was low, meaning a 1K record, the BLSB input (lsb of the memory block) would reach the output, therefore correctly implying the MSB counter output was not needed. Since the memory block to be used is expressed as a four-bit number, the chosen least significant bit implies the block is either an even or odd number. So for a 1-Kbyte record, the data may be stored in any one of the 16 available locations (ie. location 0 to location 15), depending on "BLSB". However, if a 2-Kbyte record had been selected, the lsb of the memory block would have been disregarded and the MSB counter output would have been multiplexed through. The corresponding data could only be stored in an even numbered starting location (ie. location 0, 2, 4, 6....to location 14) since "BLSB" has been disregarded. What results out of the "ADD_MUX" cell, U2, is an 11-bit number, the most significant bit being either the MSB of the counter for a 2-Kbyte record, or the LSB of the desired memory block. The 11-bits are split up so that RAM multiplexing may be incorporated. The lower 2-bits are removed from the bus so that the resulting address can be accessed by four seperate RAM modules. Since the memory block requires 4-bits, and the LSB has been included in "ADD_MUX", the remaining 3 bits are concatenated to the upper end of the resulting 9 bits of the address bus. Thus, a new bus of 12 bits has been created which will be input into the "RAM_ADD" cell, U32. The outputs from a 4-bit shift register with an input clock corresponding to the acquisition rate forms the clocking scheme of both address and data registers. The outputs are initially in a 1000 state until the enable line, controlled by EN_DEC, is brought

high. When this happens, the appropriate output lines will go high in a sequential manner at a rate corresponding to the input clock. The action of the outputs going from a low value to a high forms the input clock required by the address latch, "RAM_ADD", and by the data latch, "DATA_REG". This scheme allows both latches and RAM to operate at 1/4 the maximum clock speed, which could be quite high for waveforms with large bandwidths.

The state of the control lines for both storage and display RAMs depends upon which mode the device is in. If "CAP_DIS" is high, the device is transferring data from the storage RAM to the display memory, thus the data registers and mux must be tri-stated to avoid bus contentions. The two RAMs are both enabled, although the storage memory is in the read state while the display RAM is writing. In the state where "CAP_DIS" is low, two possibilities exist: 1) the device is in the continuous-capture mode, or 2) the device is storing single-shot events. For the first situation, both RAMs are enabled and storing data according to the state of their R/W' and output enable lines. Since both of these RAMs share the same data lines, it is necessary to isolate the memory elements by the use of a tri-state octal buffer, which is external to the gate array. If the single-shot mode is selected, the dual-port display RAM is disabled thereby tri-stating the data lines, while the storage RAM is enabled. Also, the outputs of the data bus from the A/D converter are tri-stated.

The simplified memory block diagram is shown in Figure 4.7 on the following page.

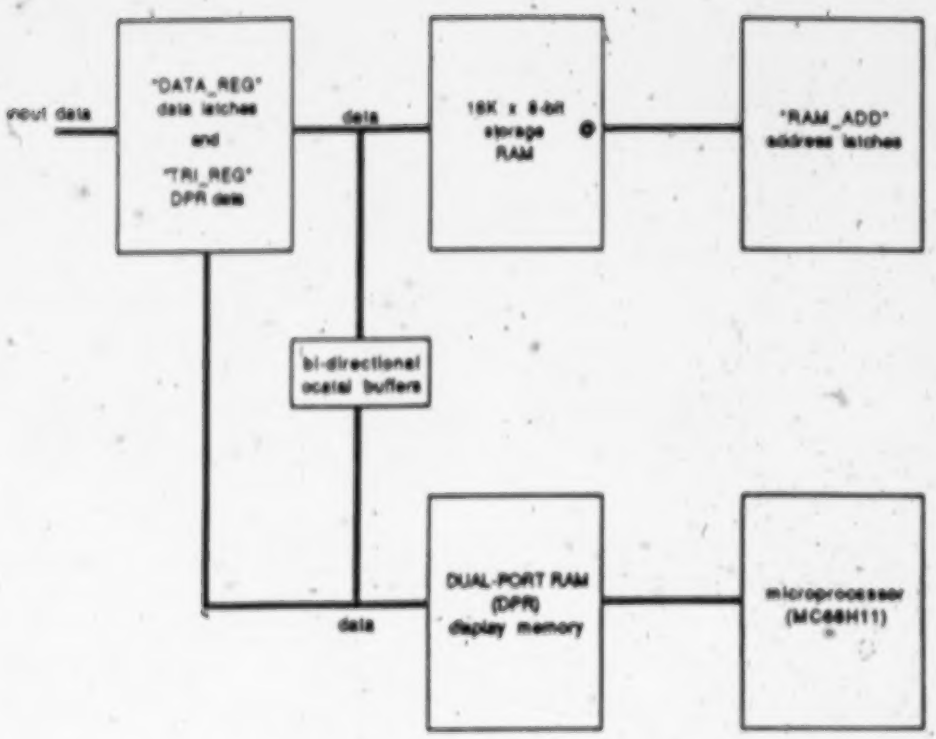


Figure 4.7

A General Block Diagram for the ADIDAT Memory Elements

4.4 Prescalers

In the clocking circuitry of this design, three types of prescalers are employed. Their purpose is to divide the high frequency system clock into several time-base ranges. The prescaler circuits PS2, PS4, and PS10 are used to divide each input clock signal by 2, 4, and 10 respectively.

Each prescaler has been constructed in a similar manner, so in order to explain the general circuit construction and operation, the variable "x" will be used, where x=2, 4, or 10. Thus, PSx was built using a chain of D-type flip flops, where the first x/2 registers are asynchronously presettable while the following x/2 flip flops are asynchronously clearable. The clock signal which is to be divided down by PSx is fed into each of the x flip flops' clock input, while each of the Q-outputs are used for the following D-inputs. The x-th flip flop Q-output is fed back to the D-input of the first flip flop.

This arrangement will produce an output clock signal with a 50% duty cycle due to the equal number of FD4 (settable) and FD2 (clearable) D flip flops. For this circuit to operate properly, it must be initialized prior to receiving the input clock signal. This is accomplished by the LD input line which causes the first x/2 flip flops to be set while clearing the last x/2 flip flops. Thus, on each input rising edge, x/2 ones are clocked through to the output after x/2 zeros. The resultant output of the PSx is a signal with a 50% duty cycle and an output frequency which is related to the input frequency by f(input)/x.

4.5 Latches

The design of this data acquisition device allows the user to store a maximum of 16 different records. For each record, there may exist a unique pretrigger value, timebase value, and a different trigger arrival point. These variables are essential in the re-construction and subsequent display of the captured digitized waveform. Therefore, a method to store these important values is required to ensure that a correct display can be produced.

To accomplish this, latches are employed to store the bit values which represent these three important variables. Pretrigger information can be expressed with 3 bits, thus a module called LATCH3 was built. Also, the point in time at which the external trigger arrives determines the amount of stored pretrigger information in the storage buffer. For this, the final value of the pretrigger counter needs to be stored when the ADIDAT receives a valid trigger. The reason for this has been explained in the data retrieval section. Thus, to store the 11-bit counter outputs, the cell named LATCH11 was used. Finally, the timebase at which the data was acquired is not a necessary variable in the re-construction process, however, it is essential during analysis of the display. Since there are 25 different timebase possibilities, 5 bits are adequate to uniquely represent each time per division selection. With this in mind, LATCH5 was constructed to hold the 5-bit number.

The n-bit latches just described were all constructed in a similar manner. Each module contains "n" D-type registers which can all be cleared by the application of a low logic level at the "CD" input of the desired latch. The registers are enabled by applying 0 volts to the "GN" input, thus, to latch a value, the GN input is taken high. For LATCH3, LATCH5, and LATCH11, all register enables

were connected together, as were the clears, so that the enabling and clearing of all n-bits occured simultaneously. These register cells have been used in the modules which provide the actual storing facility. The pretrigger values are stored using the PRE_LATCH module; the timebases are stored using the circuitry within the TIME_LATCH cell; and the pretrigger counter values are stored in the POST_LATCH module. The n-bit latches contained within each module are replicated 16 times so that there are 16 distinct registers. Each can be accessed by the memory block identifier, BLK, thereby allowing the storage of pertinant values in the same block location as the waveform. These latching circuits have been fully described in their respective sections as follows.

4.5.1 PRE_LATCH MODULE:

The PRE_LATCH module contains logic required to hold the unique 3-bit information which expresses the selected pretrigger value. The major components contained within this block includes sixteen 3-bit data registers, two 4:16 demultiplexers, and three 16:1 multiplexers. The demultiplexers are used to access one of the sixteen data register, and are controlled with the select lines connected to the memory block identifier. Storing demultiplexer U0 is enabled when storing a 3-bit value, and the output is selected by BLK3 and BLSB. The single output line is taken low while the others remain in their normally high state. This line is used to enable one of the 16 data registers which can now store the values of PRE25, PRE50, and PRE100. The "STORE" line which enabled U0 returns to its low state causing all demultiplexer outputs to go high. This action disables all registers and latches all data contained within them. To be able to use this stored information, each of the outputs from the 3-bit latch are fed into each of the three 16:1 multiplexers. The 16 PRE25 lines are inputs to multiplexer U4; the 16 PRE50 lines go to U3; and the 16 PRE100 lines are inputs to U2. Again, the

output which is to follow the input is determined by the value of the memory block specifier, which feeds the multiplexer select lines. The "RECALL" line which enables the multiplexers is always kept high in order to keep U2, U3, and U4 always active. Since the outputs of these data selectors are all inverted, the outputs need to be re-inverted using U6, U7, and U8.

To clear data contained within one of the data registers, the memory block must be specified and the "CLEAR" line must be high. Also, the "STORE" line should be low thereby disabling U0. With the clearing demultiplexer U5 now enabled, a specific register can be cleared since the outputs directly feed into the clear direct, or "CD" input of a particular latch. Thus, clearing a PRE_LATCH register defaults the pretrigger value to 0% since PRE25=0, PRE50=0, and PRE100=0.

The diagram of Figure 4.8 illustrates the organization of the PRE_LATCH data register and provides further insight into how the data is stored in this and other latches.

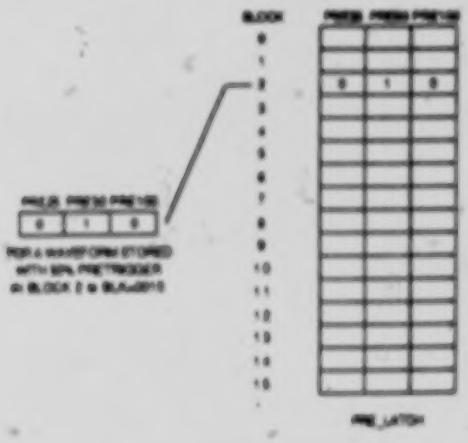


Figure 4.8

The PRE_LATCH Data Register

4.5.2 POST_LATCH MODULE:

The POST_LATCH block contains circuitry required to latch the 11-bit data which indicates the count value of the pretrigger counter. The 11-bits are the direct outputs of this counter and are latched only upon the receipt of a valid trigger. This module contains sixteen 11-bit latches, two 4:16 demultiplexers, eleven 16:1 multiplexers, and other associated glue logic. As in the PRE_LATCH, the demultiplexers control the storing and clearing process of a particular data register. The latch is selected by using the memory block identifier which forms the input control lines to both demultiplexers and output data selectors. It is important to note that latching the value from the pretrigger counter should only occur if the counter has reached its terminal count and has been re-started from the all zero state. Otherwise, the value in the data register for that particular waveform should remain to be 0. Thus, to indicate a terminal count condition, the "TC" input which arises from the PRETRIGGER block, enables the latching procedure to take place. The storing demultiplexer is disabled when the data acquisition sequence has terminated, as indicated by the low logic level on the "ACQCOMP" line. The disabling process takes all output lines of the demultiplexer to a high value, thereby disabling the data registers and latching all values within. Once latched, the value can be used in the data transfer phase, where the outputs of POST_LATCH form the preset inputs to the pretrigger counter. This has been explained in the section concerned with data acquisition and transfer.

4.5.3 TIME_LATCH MODULE:

The TIME_LATCH module contains logic required to hold the unique 5-bit value which expresses the selected timebase range. The major components contained within this block includes sixteen 5-bit data registers, two 4:16

demultiplexers, and five 16:1 multiplexers. This circuit works in exactly the same manner as the PRE_LATCH, and to some extent, like the POST_LATCH. Demultiplexer U1 is used for selecting a register where information is to be stored, while demultiplexer U0 is used to clear a particular value from a selected latch. Once again, a register is selected by using the 4-bit memory block specifier which forms the inputs of the control lines. The timebase can be uniquely specified by a 5-bit number, which arises from a 5-bit up/down counter contained within the "CLKS" circuit module. Unlike the POST_LATCH but similar to the PRE_LATCH, the desired value is latched when the input "STORE" line is brought low after being momentarily high. In this case, the "CLEAR" line should always be low. To clear a value from a data register, the "CLEAR" line is taken high momentarily while keeping the "STORE" line at a low logic level. The final outputs arise from data selectors U5 to U9 inclusive, and must be re-inverted by inverters U10 to U14 inclusive.

4.6 Counters

The three 11-bit counters contained in the ADIDAT are primary elements in the control performance of the circuit. The storing and retrieval processes are under the direct control of these counters, which are used to generate addresses for the various memory devices.

A standardized 11-bit counter was constructed using two 4-bit counters forming the 8 least significant bits, a 2-bit counter, and a 1-bit counter for the most significant bit. This design allowed address generations of either 1-Kbyte or 2-Kbyte records, depending on whether the most significant 1-bit counter was enabled or disabled. With this arrangement, the 11-bit counter required two

seperate carry-out lines; one for the full 11-bits and one for the shortened 10-bits. The carry lines were labelled CO and SHORT_CO respectively.

In the finalized design, the six carry-out lines resulting from the three counters were fed into the clock inputs of D-type flip flops. During initial design simulations, it was observed that false 0 to 1 transitions occured on both carry lines, and since the D-input was connected to +5V, a high value was latched at incorrect times. To circumvent this problem, the D-inputs were first disconnected from the +5V power source. Then, the Q0 to Q9 counter outputs were ANDed together and subsequently fed into the input of the D flip flop controlled by the SHORT_CO line. Thus, D would go high when the counter reached a value of 3FF (hex), which is approximately the same time when the SHORT_CO line goes form 0 to 1. This is due to input loading of the flip flop clock line, which then becomes slightly delayed. A similar arrangement was constructed for the flip flop activated by the CO carry line, except that the inputs to the AND gate were Q0 through to Q10. Therefore, the D-input goes high only when the count reaches a value of 7FF (hex). In both instances, the D-inputs are returned to a low value in order to avoid any further false latching conditions.

The generic 11-bit counter contained all standard input lines common to most synchronous counters. Since the counter is presettable, there are 11 preset lines with a corresponding active low LOAD line. Other inputs also include a clear (CLR) line, an enable (EN) line, a line to enable the most significant bit of the counter (EN_MSB), and a clock (CLK) line. There are 11 output lines, and, as previously mentioned, a carry-out line for 11-bit operation and a "short" carry-out line for 10-bit operation.

4.7 Timebase Generation Circuitry

In the design of this data acquisition chip, the internal clock signals required by all synchronous components are generated within the CLKS module. The circuitry contained within this cell comprises of prescalers, a latching module, a decoder, and a timebase selector.

Recall that the input analog signal is digitized at a constant rate corresponding to the sampling frequency of the analog-to-digital converter. However, the rate at which data is stored in memory is determined by the selection of a timebase. Therefore, horizontal resolution of all captured waveforms is completely determined by the selection of the timebase setting, measured in t/div.

As described in the PRESCALERS section, the system clock is divided to produce a 1-2-5 timebase sequence range, starting at 50 sec/div. and going to 10 usec/div [15]. The 24 resulting signals, plus an additional user defined external clock signal were fed into the inputs of a 32:1 decoder (named MUX32TO1). With this configuration, five lines are required to decode the signals so that only one unique clock appears at the output. The table shown in Figure 4.9 illustrates the input and output relationship for the 32:1 demultiplexer.

In order to simplify the task of setting the value on the five demultiplexer control lines, an 8-bit up/down counter (called UDC8) was built. This was done by using two 4-bit up/down counters (CUD42) from the LSI macrocells library. The five least significant bits of the counter were connected to the corresponding five control lines of MUX32TO1. The three remaining lines of the counter stayed unconnected. Two debounced keypad buttons control the number of pulses

	Input	frequency	t/div
1	00000	10 MHz	10 usec/div
2	00001	5 MHz	20 usec/div
3	00010	2 MHz	50 usec/div
4	00011	1 MHz	100 usec/div
5	00100	500 KHz	200 usec/div
6	00101	200 KHz	500 usec/div
7	00110	100 KHz	1 msec/div
8	00111	50 KHz	2 msec/div
9	01000	20 KHz	5 msec/div
10	01001	10 KHz	10 msec/div
11	01010	5 KHz	20 msec/div
12	01011	2 KHz	50 msec/div
13	01100	1 KHz	100 msec/div
14	01101	500 Hz	200 msec/div
15	01110	200 Hz	500 msec/div
16	01111	100 Hz	1 sec/div
17	10000	50 Hz	2 sec/div
18	11111	10 KHz	10 msec/div

Figure 4.9

I/O for the 32:1 Multiplexer

reaching the clock input of the up/down counter, where one button is to count up the scale, while the other provides a means to change direction to go down the scale. The outputs Q0 to Q4 from the counter generates the 5-bit input to the demultiplexer in a sequential binary order. This enables the user to go through all possible timebases in either direction, as is shown Figure 4.10.

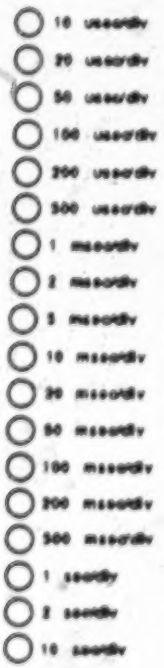


Figure 4.10

Timebase Organization

These timebases are important to know for the analysis of the re-constructed waveform. Since each of the stored data records may have been acquired using varying timebase ranges, it would be essential to store these values for future reference. Recall that each data record has a unique storage location assigned to it through the block (BLK) identifier. Therefore, by using the TIME_LATCH module described earlier, it is possible to store the 5-bit number defining the timebase of a particular waveform. This value would be stored in TIME_LATCH at the location corresponding to the storage location of the data block. With this method, by specifying the block number during re-display, both correct data and timebase will be recalled.

The inputs to the CLKS module will now be examined. In order to generate the variety of required clock signals, a system clock must be fed into the "XCLK" input of this cell. This clock is subsequently divided down by the prescaler circuitry, which needs to be initialized through the "LD" input. The lines which are required by the TIME_LATCH module are the standard "STORE", "CLEAR", "RECALL", and "BLK" inputs. For the up/down counter, enabling is accomplished by the "EN" pin, count direction is determined by the "UP" input, presetting inputs to the counter is done by a momentary low value on the "LOAD" pin, and the counter is cleared with 0 volts applied at the "CLR" input. The pulses from the keypad to increment or decrement the counter are fed into the "TIMESEL" input pin. Finally, a pin is available for a user-defined external clock, and is applied to "EXTCLK".

The output of the 32:1 demultiplexer has an inverted output, so an output inverter was required to maintain the same clock phase. For the CLKS module to

be able to drive all synchronous components in this design, high-drive inverters were utilized, providing large fan-out capabilities and small rise-time transients.

4.8 Control Circuitry for the External LED's

The purpose of the logic circuit controlling the light-emitting diodes is to provide an indicatation as to which RAM blocks have been filled with data and which remain empty. The display also informs the user whether a 1-Kbyte or 2-Kbyte record length is contained at a certain location. The circuity which controls the operation of the LED's is contained within the block labelled "RAM_LED", and will be described here in detail.

During system initialization, the "CLR" line is momentarily set to zero in order to clear all latches contained within this circuit. At the same time, the "STORE" line is taken high, indicating that data is being stored in memory, after which the circuit is ready for user enteries. The user must first decide on the desired record length of the digitized waveform, and then determine which RAM block it is to reside in. Upon selection of these variables, the three BLK3 lines, BLSB, and the MEM_SEL lines will all be set accordingly by the microprocessor, which will be interfaced with the keypad.

The circuitry is now set up to control the numbered LED's, which have been organized to assign memory blocks with either an even or odd numbering scheme. With 16-Kbytes of storage RAM available, it can be partitioned into 16 x 1 Kbyte segments, or 8 x 2 Kbyte blocks. As has been mentioned before, the memory block to be used is expressed by a four-bit number, where the least significant bit (BLSB) determines whether the block is an even or odd number. If a 1-Kbyte record has been picked, the data may be stored in any one of the 16 available locations (ie. location 0 to location 15), but for a 2-Kbyte record, the corresponding

data can only be stored in an even numbered memory location (ie. location 0, 2, 4, 6....to location 14). For a 1K storage length, the LED corresponding to the block which holds the data will be illuminated. For the situation where a 2K record is to be stored, both the corresponding even and subsequent odd LED's are illuminated, as well as the double arrow LED's. The double arrows indicate that a 2K record has been stored at those locations, as opposed to two 1K records stored at an even and odd location in succession. This can be more easily understood by examining both the LED layout arrangement and the circuit diagram for RAM_LED, given in Figures 4.11 and 4.12 on the following pages.

As seen on the circuit diagram, there are three sets of 8 registers which correspond to the 16 possible memory locations available, and for allowing arrow illumination. These have been divided such that the 8 J-K flip-flops found in U3 represent the even numbered locations whereas the other 8 J-K flip-flops found in U6 represent the odd locations. The registers found in U14 are used to turn on the triangular lights. When a particular flip-flop stores a high logic level, then its corresponding light-emitting diode is turned on. The J-inputs are fed through XOR gates from the outputs of an "even" 3:8 decoder (U7), an "odd" 3:8 decoder (U8), and a "memory length" decoder (U13). For a logic 1 to appear at the J-input, the "STORE" line must be high, indicating that data is being stored. However, if "STORE" is low, clearing is taking place and the XOR gate outputs will be inverted to a low logic level. This causes a light to turn off. The three inputs to all of the decoders arise from levels set on the "BLK3" lines. The combination of "BLSB", "MEM_SEL", and logic gates U0, U1, U2, and U19 determines which of the decoders are enabled. The J-K flip-flop which is clocked high is dependant upon whether both the output of the decoder and the "TC" (terminal count for the pretrigger counter) input are high. The value of "TC" originates from the

L.E.D. LAYOUT SCHEME :

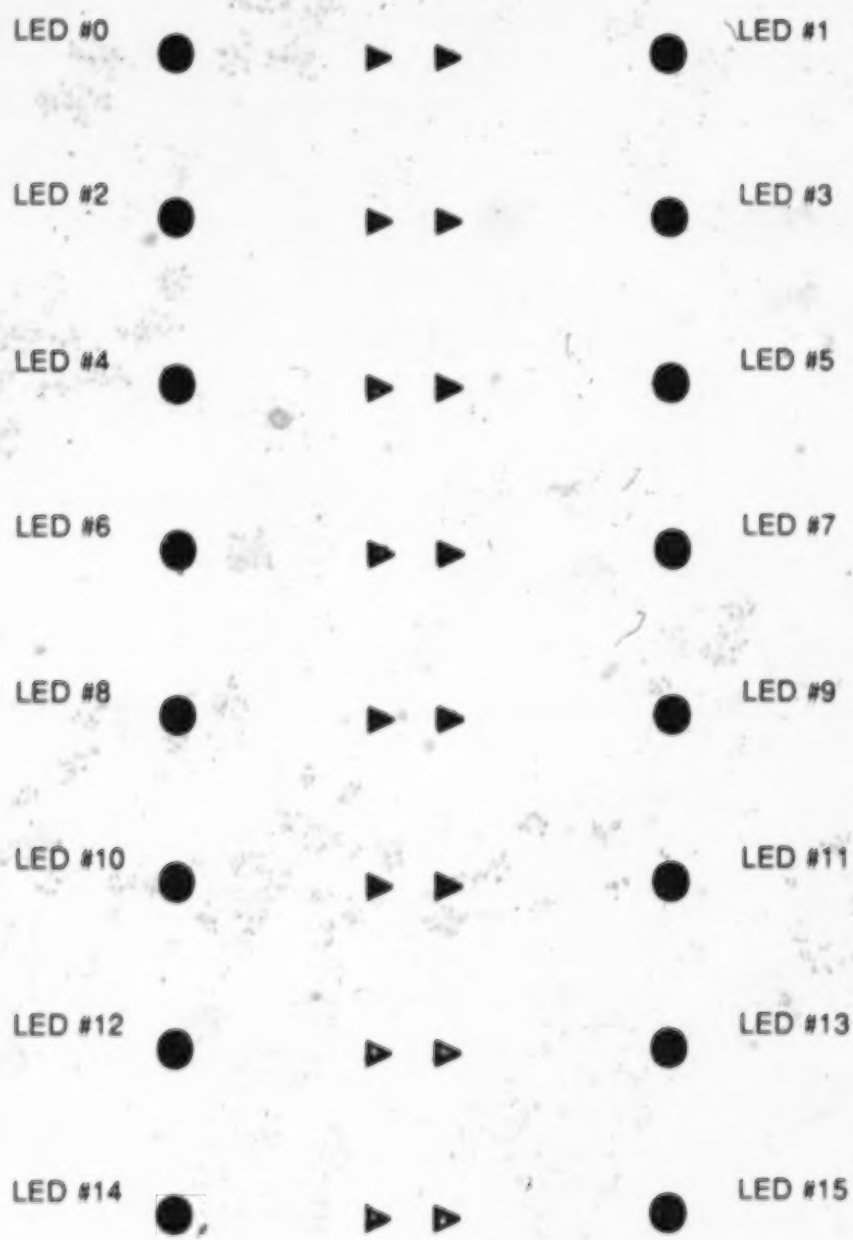


Figure 4.11

Layout of the LED's

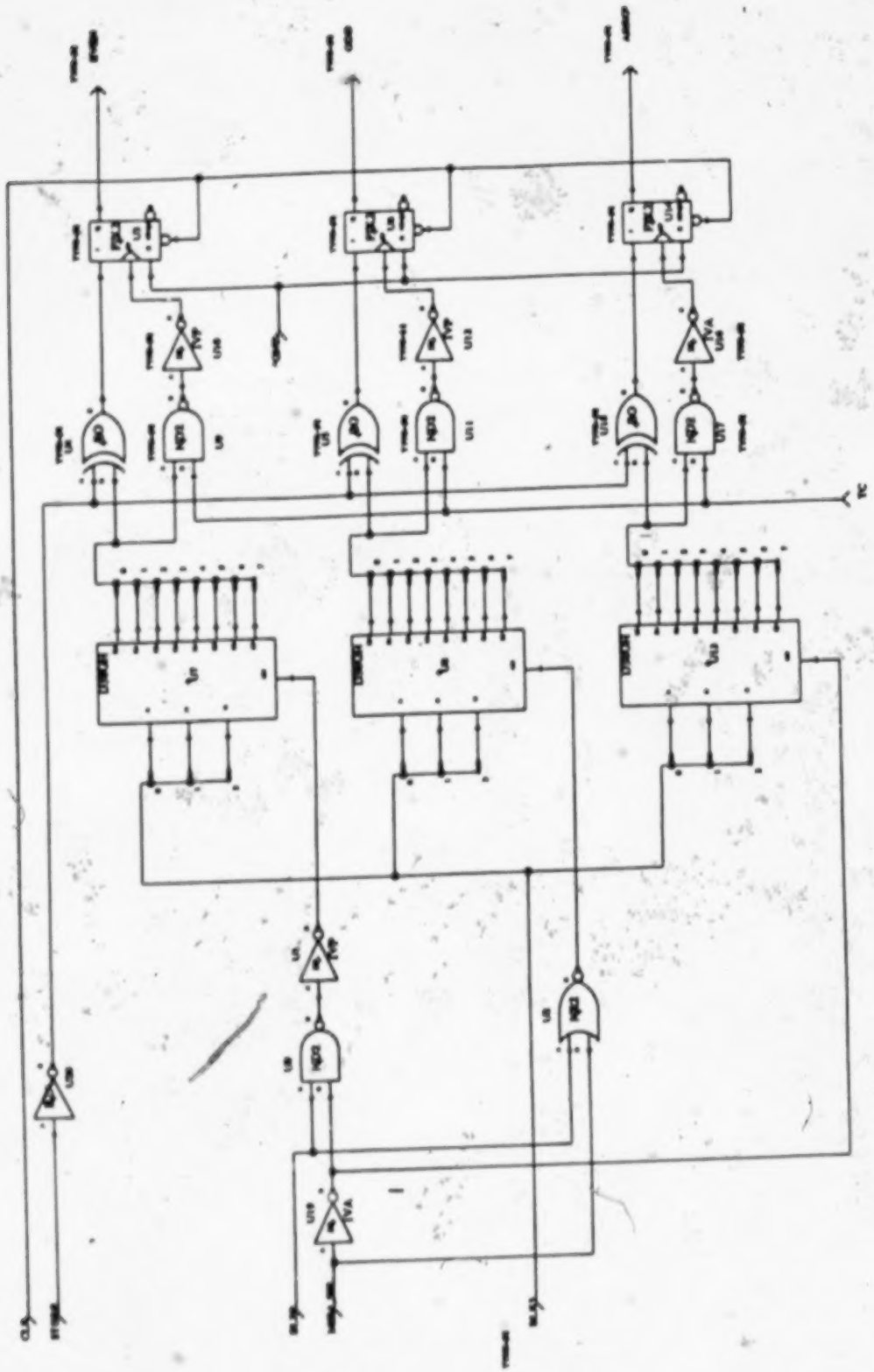


Figure 4.12

Block Diagram of the RAM_LED Module

PRETRIGGER module and goes high when 1 Kbyte of data has been stored. This is independent of the selected record length.

For the first test case, if the user wishes to store a 1K waveform in RAM block 3, MEM_SEL is set LOW to indicate a 1K record. The "BLSB" goes high and the lines to indicate the rest of the memory block are BLK3=001. After the latches have been cleared, the "STORE" line is brought high and the storage process can commence. The output of U0 is 0 thereby taking U1 high and subsequently disabling U7. However, with differing inputs to U2, the output of the NOR gate is low thereby enabling the odd decoder U8. Also, since "MEM_SEL" is high, the arrow decoder is disabled by the action of inverter U19. For "BLK3"=001, this produces a high output value for line Q1 of the odd decoder. This represents RAM block 3, and is fed through the XOR gate since "STORE" is high. Thus, the odd flip-flop input J1 has a high value ready to be stored when the pretrigger terminal count, "TC" has been reached.

In the second case where the user selects a 2K record length, the value of "MEM_SEL" is 1, so all three decoders are enabled. Thus, for "BLK3"=001, both LED lights for RAM block 2 and 3 are lit up, as well as the double arrow LED's, which indicate a 2K storage.

5. Testing the ADIDAT

5.1 Design Implementation and Procedure

Due to the familiarity and availability of LSI Logic Corporation's software package for logic design [19], this graphics tool was utilized in the design-work for this project. LSI's design system, known as The LDS System, provides the basis of a family of logic design tools. The LDS System contains interactive graphics entry tools (eg. LSED and LWAVE) and analysis tools (eg LCMP, LSIM, LVER, and LTEST) for completing a full custom chip design. With this type of design environment, once the circuit has been reliably simulated to perform the desired functions, then the design is said to be "correct" with respect to those specific situations. Therefore, to cover all possible occurences of varying situations, many simulation steps must be carried out to verify proper circuit functionality. The simulation phase of the design can be paralleled to breadboard testing, where in both cases, circuit correctness is verified. The steps required to complete a logic design are described here in detail, and correspond closely to the descriptive flowchart shown in Figure 5.1.

DESIGN STEPS

STEP 1:

With the Logic Schematics Editor (LSED), the schematic design is entered, from which LSED automatically generates and checks the netfiles which are input to the LDS netfile compiler program LCMP.

DESIGN STEPS:

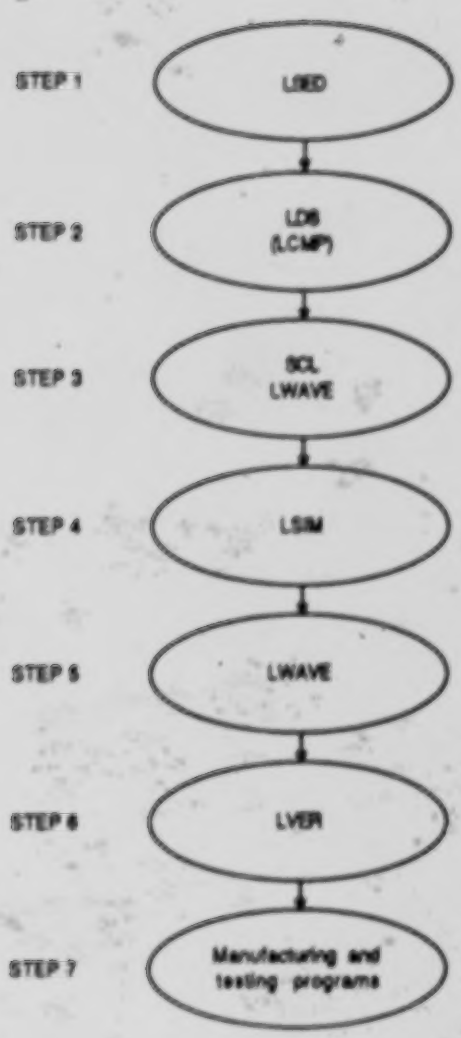


Figure 5.1

Steps Required to Complete a Logic Design

STEP 2:

The netfile compiler (LCMP) is executed to compile all part descriptions contained within the netfiles.

STEP 3:

After compilation, the network is simulated using the Simulation Control Language (SCL) which forms the input to the LDS simulation program, LSIM. SCL files can be generated by writing a simulation program, or by using the LWAVE graphics tool with user-created input waveform patterns. The simulation verifies circuit performance, so if an error in design occurs, modifications must be made to correct the problem (ie. repeat step 1).

STEP 4:

Execution of LSIM generates the simulation results of the design. Also, the program automatically links all network modules with LLINK, and creates a delay file using LDEL.

STEP 5:

If LWAVE was used for SCL file generation, then the simulation output can be viewed using LWAVE.

STEP 6:

Next, the LDS verification tool (LVER) is executed which checks for electrical feasability of interconnections between circuit modules. Also, it gives the number of gates required by the design.

STEP 7:

After the verification program has been run, the LDS layout programs are executed, and after extensive testing, the chip design is sent to be manufactured and re-tested.

The LSED schematics editor allows the user to build the modules of the logic design using a standard library of parts. Schematics can be edited using LSED in much the same way that a text-editor edits textual information. The LSED program is organized into levels allowing for a hierarchial organization of the circuit design. For each level in a design, there is a symbol level and its corresponding network level. The exception to this is the top, or first level, where no distinct symbol level exists. The level structure has been depicted below in Figure 5.2:

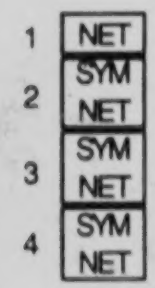


Figure 5.2

Hierarchial Level Structure

In the hierarchial design, the top level consists of a single comprehensive part. This top-level part consists in turn of one or more sub-parts, which themselves may contain sub-parts. LSI Logic requires that every logic design done by means of the LSED program be hierarchially organized, and an example of this has been shown in Figure 5.3.

With respect to the design of this project, the hierarchial structure of the ADIDAT chip appears as shown in Figure 5.4.

The top-level network was labeled "FINAL1", and comprises of all modules and sub-parts found within the design. Level 2 consists of the circuitry found within the "FINAL1" module, thus the level 2 symbol is again "FINAL1". However, the network level comprises of control logic and sub-cells, namely "CLKS", "CIRCUIT1", "PRE_LATCH", and "COUNT4F". Proceeding further into this hierarchial tree structure shows that many sub-circuits are contained within level 3. Depending upon which level 2 module was selected, a particular level 3 symbol and network will be given. For example, choosing the level 2 cell labeled "CIRCUIT1", the level 3 symbol would be "CIRCUIT1" with a network containing the modules "POST_LATCH", "TESTER", "RAM_LED", and "SELECTOR". Each of these modules has been explained in an earlier section, except for "TESTER", which does not explicitly perform a function on its own. It was built only for convenience since the "TESTER" cell consists of important sub-parts "PRETRIGGER" and "MEM_CTR", which are both fully explained in subsequent sections. It should also be noted that some of the labels attached to various circuits are not representative of the function of that cell, but rather are only names assigned during the circuit design and simulation phase.

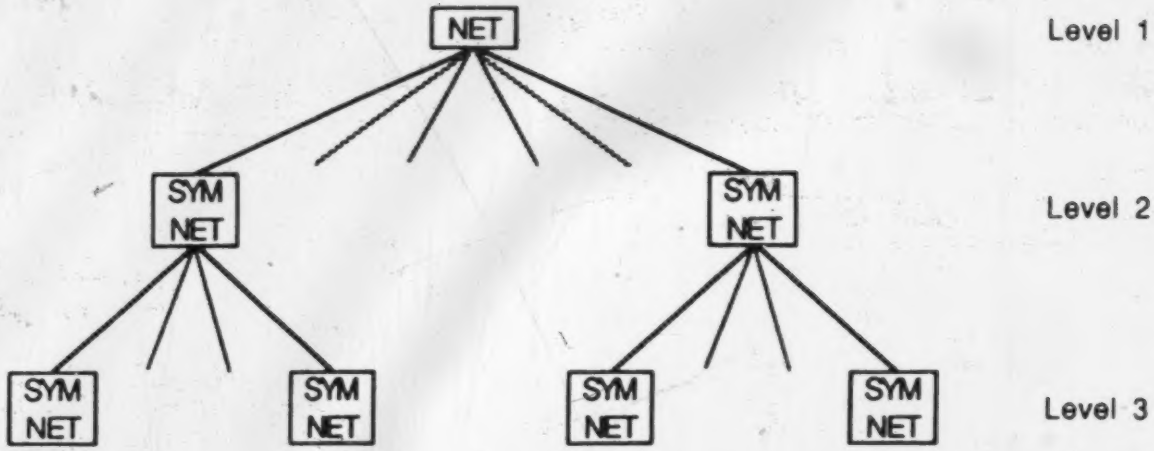


Figure 5.3

An Example Showing a Heirarchial Design

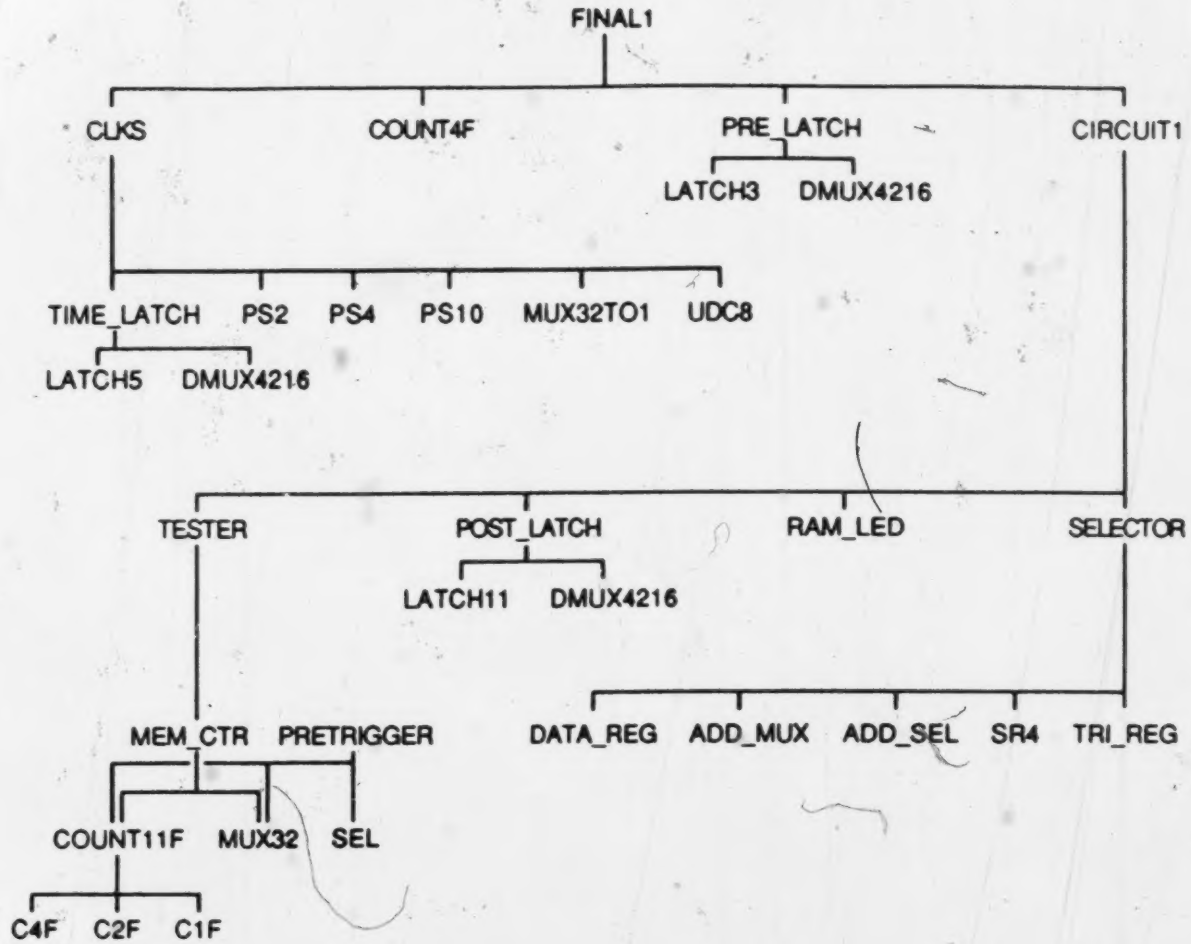


Figure 5.4

The Hierarchy of the ADIDAT Module

5.2 Simulation Techniques

In order to test the functionality of the design, comprehensive simulations must be performed to ensure proper circuit operation. With the use of LSI Logic's LDS simulation software [19], the design can be verified for all aspects of logic performance. This also allows the testing of the circuit in situations comprising of a variety of operating voltages, temperatures, as well as gate delay standards.

The degree to which a design is simulated rendering the expected results gives an indication of the correctness and validity of the circuit. The results must confrom exactly to what the designer wanted or expected of the circuit. Once this criterion has been satisfied, the design can be deemed as being functionally correct for the given test conditions, and the necessary steps for chip fabrication may be initiated.

Since the LDS system supports a hierarchial design structure, the total circuit should comprise of lower level sub-modules. The advantage of this method is that each cell is stand-alone, meaning it has its own unique netfile. Thus, each sub-cell can be tested independently for proper operation before being connected to the next higher hierarchial level.

Along with circuit functionality, the LSIM program also generates a delay synopsis for all outputs contained within the sub-cell. This information gives an indication of the fan-out, since the delay list measures how long a low-to-high or high-to-low transition takes for a given output. Thus, for a large fan-out, more output drive is required to provide the proper logic levels to each of the successive inputs. The generated delay list should therefore be examined for fan-out problems as LSI Logic's performance standard requires all rise and fall times to be less than

5ns. for nominal condition testing (that is, at temp = 25C, and voltage = 5 volts). It is apparent that the simulation output should be used to test for proper logic performance, and to make any logic corrections if necessary. Also, the delay synopsis is required to ensure proper drive capabilities, and corrections can be made by the addition of buffer drivers. Although delay errors may have been eliminated in a sub-module, rise and fall time violations may again arise for the "corrected" cell after it has been connected in a higher-level hierarchy. Therefore, all levels of the network should be tested for functionality and fan-out.

With this kind of methodology, the top-most level of the design should be flawless in both circuit performance and delay condition analysis. This procedure was carried out with the design of the data acquisition chip. In the following section, the procedure for simulating the ADIDAT will be discussed.

5.3 Simulation Results

With LSI Logic Corporation's "Logic Design System", chip level digital logic can be compiled, linked, simulated, and verified using the available LDS programs [19].

A circuit design is first described by schematic capture using the LSED graphics editor. When various circuit components are to be saved, LSED generates the required netlists of all circuit modules contained within the design. In order to simulate the logic design with the LDS simulation tool, it is necessary to execute commands that can compile, link, and generate delay information from the netfiles. First, the LCMP command translates all cell netfiles into a format suitable for linking with each other, and with the LSI Logic libraries. Upon execution of the LLINK command, the compiled netfiles are linked with the library macrofunctions, macrocells, and other circuit modules which define the total logic

design. Next, delay information such as rise and fall times for network components, is determined using LDEL. This information reflects circuit performance for various operating parameters like temperature, operating voltage, and process-variation values. Finally, simulation of the network design is accomplished by invoking the LSIM command. The simulator reads and executes the files containing the linked network description, the generated delay information, and the simulation control language (SCL). The LSIM command generates simulation results for the logical values assigned to specific signals within the network. As the simulation proceeds, results are generated as specified by the printing and formatting commands issued in the SCL program. Further information on LSI Logic's simulation control language can be found in reference [19].

Various simulations were designed to show the data handling capabilities of the ADIDAT chip for a wide range of situations and operating conditions. It was therefore necessary to simulate the incoming data stream of the digitized waveform from the A-to-D converter output to the ADIDAT input. This was accomplished by creating a "pattern" file which consisted of 255 8-bit numbers representing the instantaneous amplitude of the analog input signal. Since the A/D converter is set to operate at a constant conversion rate of 20 MHz, each of the 255 8-bit numbers is read into the ADIDAT, in succession, every 50 nanosecond. The pattern file scheme can be easily incorporated with the simulation control language (SCL), and is supported by the LDS simulator program, LSIM. As can be seen from the "FINAL1" schematic, the input data enters the ADIDAT on the 8-line data bus labelled DATA_IN.

To simplify the simulation process, it is assumed that the input waveform to be digitized is a sawtooth signal, as shown in Figure 5.5 on the following page.

This simplification is used to allow the reader the ability to read the simulation outputs more clearly. However, it should be emphasized that any input waveform can be used to simulate the digitized signal. With a sawtooth waveform, the simulated digital signal can be simply constructed as an increasing 8-bit binary number, starting at 00000000 (00 hex) and ending at 11111111 (FF hex). The ramp generated by the 255 data points is repeated again and again, which produces the resulting digitized sawtooth function. This scheme can also provide an easy check on the reconstruction simulation program to ensure that a sawtooth waveform is regenerated from the original data.

Numerous simulation results were obtained, each dealing with a different capture or display option. However, each simulation result can be read in exactly the same manner since all have been set to follow the same output formatting. One such output listing has been given in Appendix A at the end of this document.

The simulation demonstates the single-shot capture of a waveform in a 1 Kbyte memory block. The storage occurs in block 9 using the ROLL mode at 10 MHz. The trigger arrives after the pretrigger requirements have been satisfied. All of the simulation results show the time at which any of the displayed variables changes, and is shown in the first column. It should be noted that the given time steps are stated in nanoseconds x 10, thus for time=136, this would correspond to 13.6 ns. after the simulation had commenced.

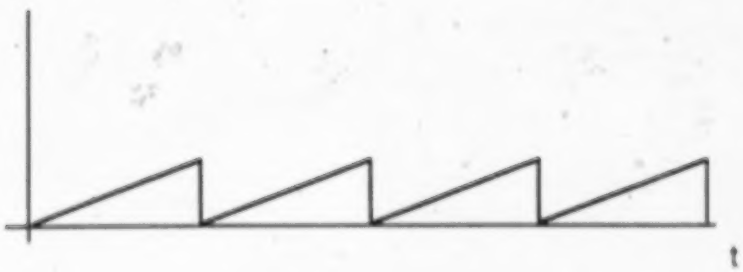


Figure 5.5

The Simulated Input Waveform

The second column in the formatted output shows the internal clock frequency, and is denoted by "INTCLK". The third column displays the arrival of a valid trigger, and is called "TRIGGER". The following three columns give the output of the three 11-bit programmable counters in hexadecimal format. The "PRE_CNTR" displays the outputs from the pretrigger counter; the "POST_CNTR" shows the outputs from the posttrigger counter; and the "MEM_CNTR" column displays the count value from the memory counter. Recall that the conter in the MEM_CTR block is only active during a data transfer phase, that is, when transferring data from the storage RAM into the dual-port display RAM. The next four columns which follow deal with the signals relating to only the dual-port RAM (or DPR). These include the active-low chip enable line (DPR_ENAB); the active-low write enable (DPR_WE); the storage RAM address (DPR_ADDR); and finally, the storage RAM data (DPR_DATA). As with the display memory, the storage RAM has its corresponding chip enable (RAM_ENAB) and RAM address (RAM_ADDR) signals. However, because RAM multiplexing is being utilized in this design, four distinct storage RAM modules are required. Thus, 2 sets of four columns are used to indicate the data being stored in each RAM location (DATA0, DATA1, DATA2, and DATA3) given by the corresponding address, governed by the RAM_ADDR column. For each of the 4 RAM cells, there are four distinct

write enables which select which RAM out of the four is currently writing data from the data registers. These lines are all active-low so they remain normally high when in the READ state. Finally, the last three columns, designated by EVEN, ODD, and ARROW, are used in conjunction with the display LED's. The hex number which results determines which light-emitting diode is turned on and which are off. This relates to which RAM block is filled with data, and has been discussed earlier.

With the simulation output format now known, it is now possible to examine the simulation results for the case described earlier. In order to conserve space in this document, the simulation has been shortened by editing the output file. This was neccessary since normally, the entire simulation would take up around 500 pages to display all salient information. Since all of the results in this single simulation displays a repetitve nature, this generalization can easily be made without the loss of too much information.

With reference to the simulation results shown in the appendix, the start is made at t=0, where the ADIDAT is in the initialization state. Here, settings for the mode, pretrigger requirement, and storage block value for either a capture or display sequence can be made. It is apparent that the circuitry remains in a relatively unstable state until t=43.2 ns., at which point the outputs become defined. For the selected ROLL mode of operation with single-shot capturing, the dual-port RAM (DPR) is disabled and therefore the DPR data lines are tri-stated. This is indicated on the output listing by an undefined state value for DPR_DATA. Similarily, since this simulation deals with a capture phase rather than a data transfer sequence, the memory counter, or MEM_CTR, is disabled.

At t=300 ns., the "clear variables" line is brought high (CLR_VAR=1) thereby disabling any further clearing of the variable latches. After t=350 ns., the block identifier is loaded into the circuitry, thus the output BLOCK displays a "9" signifying the selected block value. At t=400 ns., the desired timebase is selected. This has not been indicated on the output listing, however, the variable "TIMSEL" is brought momentarily high to increment the 5-bit up/down counter from 1F (hex) to 00 (hex). This "time value" corresponds to a 10 MHz capture rate. Next, the "STORE" line is momentarily set to a high value to load the latches with the pretrigger value and the selected timebase.

Prior to execution, it is important to note that the four data latches, DATA0 to DATA3, contain zeros and their respective write enable lines are all high. The RAM address shown in the output listing as RAM_ADDR is held at 000. Also, the LED outputs EVEN, ODD, and ARROW are all zeros. Similarily, the pretrigger and posttrigger counters remain at their all zero state since no clock is present prior to execution.

Finally, at t=1000 ns., the EXECUTE line is brought low thereby initiating the acquisition phase. It can be seen that the internal clock is running and that the digitized input signal (sawtooth waveform) is incrementing every 100 ns. With the clock now active, both the pretrigger and posttrigger counters begin to increment at a rate of 10 MHz. The input data is available to the data registers to be latched on a low logic level on the write enable (WE) lines. Also, after a slight delay of 61 ns., the RAM address contains the block identifier. For address 900 (hex) and t=1148 ns., data value 01 is latched into DATA0; at t=1248 ns., data value 02 is latched into DATA1; t=1348 ns., data value 03 is latched into DATA2; and at t=1448 ns., data value 04 is latched into DATA3. The RAM address then

changes to 901 (hex) and the process is repeated for different data values. Before the data registers are over-written, the RAM should read the latched data.

At t=52,248 ns., the pretrigger data requirements are met and the pretrigger counter re-starts at the all zero state. However, the posttrigger counter is halted at 200 (hex), which corresponds to 50% of a 1K record. The pretrigger counter will continue to increment until a valid trigger is received. This has been arbitrarily set to occur at t=60,000 ns. When this happens, the pretrigger counter is frozen, and the posttrigger counter begins counting from 200 (hex) upto 400 (hex). At t=111,086 ns., the LED indicators are set to display a stored waveform in block location 9. The acquisition phase is now complete, but what is not shown on the simulation output is that the value of the pretrigger counter is held in the POST_LATCH counter register for data re-displaying purposes.

The ADIDAT is now ready for another data capturing or data transfer situation.

6. Conclusions

Having described the details of the ADIDAT logic array, the usefullness of this device as a principle module in digital storage oscilloscope systems is quite evident.

The final design consists of 40 input lines and 93 output lines. In order for the microprocessor to set the control lines, it may be necessary to externally multiplex the inputs. The output data lines should be tied directly to the input data bus of each storage RAM module, while the address lines are connected to each of the 4 RAM modules. This can be seen in the system block diagram given earlier. With the microprocessor and keypad, routines can be developed to facilitate scrolling and zooming functions of the displayed waveform. Also, programmes can be implemented which would display grids, time-steps, voltages, pretrigger constraints, and memory blocks on the screen.

The final circuit meets all of the designers criteria in performance and operation, as set out in sections 2.4 and 2.6. This has been accomplished through the analysis of numerous comprehensive design simulations. Thus, according to LSI Logic Corporations' design strategy, the design can be deemed to be functionally correct and can therefore be sent for further analysis leading to I.C. fabrication. Once the custom-designed chip has been completed, it can be utilized in any quasi real-time data handling application. However, its' primary use as a component in DSO systems, along with memory and A-D/D-A converters, will enable the user to adapt an existing analog scope into a hybrid oscilloscope. This has the advantage of making use of the existing triggering circuitry as well as the oscilloscope screen. The ADIDAT system can also be used in conjunction with a monitor and an appropriate CRT controller thereby avoiding the neccessity of

having an analog oscilloscope. Since the final instrument would make use of a custom designed integrated circuit, the resulting component count will be drastically reduced with the benefit of reliability and ease of maintenance. Thus, employing the ADIDAT would be a cheap and effective method of obtaining the characteristics desired in a digital storage oscilloscope.

7. References

1. Staff Writer, "Digital Storage Oscilloscope", Electronic Design, Vol. 33, No. 6, March 14, 1985, pp.205-216.

2. A. Tegan, J. Wright, "The Digital Scope Alternative", Canadian Electronics Engineering, Vol. 29, No. 1, February, 1985, pp.12-19.

3. L. Teschler, "Muscling Out Analog Technology", Machine Design, Vol. 57, No. 6, Mar. 21, 1985, pp.97-103.

4. A. Lechner, H. Jessner, R. Petschacher, "Data Capture Matches Flash Converter Speeds", Electronic Design, Vol. 36, No. 8, Mar. 31, 1988, pp.101-112.

5. J.W. Read (Editor), "Gate Arrays - Design Techniques and Applications", McGraw-Hill Book Company, 1985.

6. G. Bostock, "Programmable Logic Devices", McGraw-Hill Book Company,1988.

7. J. Helfferich, E. Kruisdijk, "Component Integration in Oscilloscopes", Electronics and Wireless World, Vol. 93, No. 1608, October 1986, pp.57-59.

8. H.V. Malmstadt, C.G. Enke, S.R. Crouch, "Electronics and Instrumentation for Scientists", The Benjamin/Cummings Publishing Company, Inc., Feb., 1981.

9. P. Horowitz, W. Hill, "The Art of Electronics", Cambridge University Press, Cambridge, 1980.

10. Staff Writer, "Digital Storage Oscilloscopes", Electronics and Wireless World, Vol. 90, No. 1579, April, 1984, pp.63-67.

11. M. Allen, J. Sorden, "Digitizing Oscilloscopes Finesse Analog Signals With Tools and Techniques", Electronic Design, Vol. 33, No. 30, Dec. 26, 1985, pp.83-87

12. S. Lang et al., "Hybrid Instruments force an Alliance Between Analog and Digital Worlds", Electronic Design, Vol. 33, No. 2, Jan. 24, 1985, pp.117-152.

13. "Bandwidth And Sampling Rate In Digitizing Oscilloscopes", Hewlett-Packard Application Note No. 344, April 1986.

14. H. Landau, "Sampling, Data Transmission, and the Nyquist Rate", Proc. IEEE, 55/10, October 1967.

15. J. Wang, D. Murphy, "Add Waveform Storage To Your Oscilloscope", Popular Electronics, Vol. 20, No. 4, April 1982, pp.42-52.

16. M. Koen, "Comparing ADC Architectures is a Designers Best Bet", Electronic Design, Vol. 35, No. 2, Jan. 22, 1987, pp.119-122.

17. B. Milne, "Digital Oscilloscopes", Electronic Design, Vol. 34, No. 3, Feb. 6, 1986, pp.98-106.

18. "Linear I.C. Data Book, Volume 2", Samsung Semiconductor and Telecommunications, September 1987, pp.521-548.

19. "LSI Logic Corporation's Schematic Capture and Simulation Control Language", LSI Logic Corp., 1986.

20. "Nicholet 3091 Digital Oscilloscope Operating Manual", Nicholet Instrument Corporation, pp.1-1 to 6-30.

21. "High-Speed CMOS Data Book", Integrated Device Technology, Inc., January 1988, pp.5.16-5.28.

22. R. Walker, D. Thomas, "A Model of Design Representation and Synthesis", Proc., 22nd Design Automation Conference, 1985, pp. 453-459.

APPENDIX A

SIMULATION OUTPUT FILE

Sat May 6 22:56:16 1989 LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987) PAGE: 12
NET: FINAL1 SOFTWARE LOGIC SIMULATOR COLUMN: 1
SIMULATION RESULTS

TIME	INTCLK	TRIGGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADDR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADDR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
0 :																									

******* THIS SIMULATION DEMONSTRATES THE SINGLE-SHOT CAPTURE OF A WAVEFORM IN A 1K MEMORY BLOCK WITH 50% PRETRIGGER. STORAGE IS IN BLOCK 9 USING THE ROLL-MODE AT A 10MHZ SAMPLING RATE. THE TRIGGER ARRIVES AFTER THE PRETRIGGER TERMINAL COUNT HAS BEEN REACHED. *******

0 :

INITIALIZATION:

RESET=0
CLR_VAR=0
XCLK=0
EXTCLK=0
TIMESEL=0
UPDOWN=1
BLKLOAD=0

0 :

CONTROL LINE STATUS:

ROLL=0	STORE=0	PRE25=0	LOCK_DPR=1	BLSB=1
REFRESH=1	CLEAR=0	PRE50=1	REL_DPR=1	BLK3.0=0
CONT_CAPT=1	RECALL=0	PRE100=0	HOLD=1	BLK3.1=0
SINGLE_SHOT=0	LONG_REC=0	MEM_SEL=0	REL=1	BLK3.2=1
RESET_MODE=1	STORELEV=0	CAP_DIS=0		

0 :

INPUT STATUS:

TRIGGER=0
EXECUTE=1

0C:	X	0	01	--	---	---	X	---	--	X	X	---	--	X	--	X	--	X	--	X	--	--	--	-	--
18C:	X	0	01	---	---	---	X	---	--	X	X	---	00	X	00	X	00	X	00	X	--	--	--	-	--
21C:	X	0	01	---	---	---	X	---	--	X	X	000	00	X	00	X	00	X	00	X	--	--	--	-	--
29C:	X	0	01	---	---	---	X	---	--	X	X	000	00	X	00	X	00	X	00	X	00	00	00	-	--
39C:	X	0	01	---	---	---	X	---	--	X	1	000	00	X	00	X	00	X	00	X	00	00	00	-	--

Sat May 6 22:56:16 1989
NET: FINAL1

LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987)
SOFTWARE LOGIC SIMULATOR
SIMULATION RESULTS

TIME	INTCLK	TRIGGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADDR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADDR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
104C:	X	0	01	---	---	---	X	---	--	X	1	000	00	X	00	X	00	1	00	0	00	00	00	-	--
105C:	X	0	01	---	---	---	1	---	--	X	1	000	00	X	00	X	00	1	00	0	00	00	00	-	--
106C:	X	0	01	---	---	---	1	---	--	X	1	000	00	X	00	1	00	1	00	0	00	00	00	-	--
110C:	X	0	01	---	---	---	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	-	--
120C:	X	0	01	---	---	---	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	-	--
121C:	X	0	01	---	---	---	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	-	--
125C:	X	0	01	---	---	---	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	-	--
127C:	X	0	01	---	---	---	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	--
157C:	0	0	01	---	---	---	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	--
199C:	0	0	01	---	---	---	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	--
201C:	0	0	01	---	---	---	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	--
203C:	0	0	01	---	0--	---	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	--
204C:	0	0	01	---	0--	---	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	--
205C:	0	0	01	---	0--	0--	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	--
207C:	0	0	01	---	0--	0--	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	--
209C:	0	0	01	---	0-0	0-0	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	--
210C:	0	0	01	---	000	000	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	--
220C:	0	0	01	---	000	000	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	--
223C:	0	0	01	---	000	000	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	--
224C:	0	0	01	0--	000	000	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	--
226C:	0	0	01	0--	000	000	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	--
228C:	0	0	01	0-0	000	000	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	--
229C:	0	0	01	000	000	000	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	--
244C:	0	0	01	000	000	000	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	0-
273C:	0	0	01	000	000	000	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	0-
277C:	0	0	01	000	000	000	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	0-
283C:	0	0	01	000	000	000	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	0-
296C:	0	0	01	000	000	000	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	00
298C:	0	0	01	000	000	000	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	00
303C:	0	0	01	000	000	000	1	---	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	00
304C:	0	0	01	000	000	000	1	0--	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	00
305C:	0	0	01	000	000	000	1	0-0	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	00
306C:	0	0	01	000	000	000	1	0-0	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	00
308C:	0	0	01	000	000	000	1	0-0	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	00
309C:	0	0	01	000	000	000	1	000	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	0	00
432C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	0	00

Sat May 6 22:56:16 1989
NET: FINAL1

LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987)
SOFTWARE LOGIC SIMULATOR
SIMULATION RESULTS

TIME	INTCLK	TRIGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADDR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADDR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
3000 : CLR_VAR-I																									
3153C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	-	00
3155C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	-	00
3241C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	-	-0
3268C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	-	--
3272C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	-	--
3280C:	0	0	01	000	000	000	1	000	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	-	--
3295C:	0	0	01	000	000	000	1	000	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	-	--
3500 : RESET-1 BLKLOAD-0																									
3637C:	0	0	01	000	000	000	1	000	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	-	--
3639C:	0	0	01	000	000	000	1	000	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	9	--
3702C:	0	0	01	000	000	000	1	000	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	9	0-
3729C:	0	0	01	000	000	000	1	000	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	9	0-
3733C:	0	0	01	000	000	000	1	000	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	9	0-
3739C:	0	0	01	000	000	000	1	000	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	9	0-
3752C:	0	0	01	000	000	000	1	000	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	9	00
3888C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	9	00
4000 : CONTROL LINE STATUS: STORE-1																									
4226C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	9	10
4227C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	9	18
4231C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	9	1c
4239C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	9	1e
4254C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	9	1f
4269C:	0	0	01	000	000	000	1	000	--	1	1	000	00	1	00	1	00	1	00	0	00	00	00	9	1f
4351C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	9	1f
4766C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	9	0f

Sat May 6 22:56:16 1989 LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987) PAGE: 15
NET: FINAL1 SOFTWARE LOGIC SIMULATOR COLUMN: 1
SIMULATION RESULTS

TIME	INTCLK	TRIGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADDR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADDR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
4771C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	9	03
4777C:	0	0	01	000	000	000	1	000	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	9	01
4789C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	9	01
4790C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	9	00
4830C:	0	0	01	000	000	000	1	000	--	X	1	000	00	1	00	1	00	1	00	0	00	00	00	9	00
4842C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	9	00
4903C:	0	0	01	000	000	000	1	000	--	1	1	000	00	1	00	1	00	1	00	0	00	00	00	9	00
4926C:	0	0	01	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	00	00	9	00

5000 :

CONTROL LINE STATUS:
BLKLOAD=1
STORELEV=1

5500 :

CONTROL LINE STATUS:
STORE=0
RECALL=0

10000 : THE 20 MHz. EXTERNAL CLOCK IS ENABLED

10000 :

INPUT:
EXECUTE=0

10321C:	0	0	01	000	000	000	1	000	--	0	0	000	00	1	00	1	00	1	00	0	00	00	00	9	00
10423C:	0	0	01	000	000	000	1	000	--	1	0	000	00	1	00	1	00	1	00	0	00	00	00	9	00
10484C:	1	0	01	000	000	000	1	000	--	1	0	000	00	1	00	1	00	1	00	0	00	00	00	9	00
10603C:	1	0	01	000	000	000	1	000	--	1	0	000	00	1	00	1	00	1	00	1	00	00	00	9	00
10608C:	1	0	01	001	000	000	1	000	--	1	0	000	00	1	00	1	00	1	00	1	00	00	00	9	00
10612C:	1	0	01	001	000	000	1	000	--	1	0	000	01	1	00	1	00	1	00	1	00	00	00	9	00
10614C:	1	0	01	001	000	000	1	000	--	1	0	900	01	1	00	1	00	1	00	1	00	00	00	9	00
10615C:	1	0	01	001	000	000	1	000	--	1	0	900	01	0	00	1	00	1	00	1	00	00	00	9	00

Sat May 6 22:56:16 1989 LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987) PAGE: 16
NET: FINAL1 SOFTWARE LOGIC SIMULATOR COLUMN: 1
SIMULATION RESULTS

TIME	INTCLK	TRIGGER	SIGNAL	PRE $\overline{CNTR}$	POST $\overline{CNTR}$	MEM $\overline{CNTR}$	DPR $\overline{ENABLE}$	DPR ADDR	DPR $\overline{DATA}$	DPR $\overline{WE}$	RAM $\overline{ENABLE}$	RAM ADDR	DATA 0	WE 0	DATA 1	WE 1	DATA 2	WE 2	DATA 3	WE 3	EVEN	ODD	ARROW	BLOCK	TIME $\overline{VAL}$
10616C:	1	0	01	001	001	000	1	000	--	1	0	900	01	0	00	1	00	1	00	1	00	00	00	9	00
10665C:	1	0	01	001	001	000	1	001	--	1	0	900	01	0	00	1	00	1	00	1	00	00	00	9	00
10895C:	1	0	01	001	001	000	1	001	--	0	0	900	01	0	00	1	00	1	00	1	00	00	00	9	00
10954C:	0	0	01	001	001	000	1	001	--	0	0	900	01	0	00	1	00	1	00	1	00	00	00	9	00
11000 :																									
INPUT: EXECUTE=1																									
11423C:	0	0	01	001	001	000	1	001	--	1	0	900	01	0	00	1	00	1	00	1	00	00	00	9	00
11484C:	1	0	02	001	001	000	1	001	--	1	0	900	01	0	00	1	00	1	00	1	00	00	00	9	00
11592C:	1	0	02	000	001	000	1	001	--	1	0	900	01	1	00	1	00	1	00	1	00	00	00	9	00
11606C:	1	0	02	002	000	000	1	001	--	1	0	900	01	1	00	1	00	1	00	1	00	00	00	9	00
11609C:	1	0	02	002	000	000	1	001	--	1	0	900	01	1	02	1	00	1	00	1	00	00	00	9	00
11612C:	1	0	02	002	000	000	1	001	--	1	0	900	01	1	02	0	00	1	00	1	00	00	00	9	00
11614C:	1	0	02	002	002	000	1	001	--	1	0	900	01	1	02	0	00	1	00	1	00	00	00	9	00
11663C:	1	0	02	002	002	000	1	003	--	1	0	900	01	1	02	0	00	1	00	1	00	00	00	9	00
11669C:	1	0	02	002	002	000	1	002	--	1	0	900	01	1	02	0	00	1	00	1	00	00	00	9	00
11895C:	1	0	02	002	002	000	1	002	--	0	0	900	01	1	02	0	00	1	00	1	00	00	00	9	00
11954C:	0	0	02	002	002	000	1	002	--	0	0	900	01	1	02	0	00	1	00	1	00	00	00	9	00
12423C:	0	0	02	002	002	000	1	002	--	1	0	900	01	1	02	0	00	1	00	1	00	00	00	9	00
12484C:	1	0	03	002	002	000	1	002	--	1	0	900	01	1	02	0	00	1	00	1	00	00	00	9	00
12588C:	1	0	03	002	002	000	1	002	--	1	0	900	01	1	02	1	00	1	00	1	00	00	00	9	00
12605C:	1	0	03	002	002	000	1	002	--	1	0	900	01	1	02	1	03	1	00	1	00	00	00	9	00
12608C:	1	0	03	003	002	000	1	002	--	1	0	900	01	1	02	1	03	0	00	1	00	00	00	9	00
12616C:	1	0	03	003	003	000	1	002	--	1	0	900	01	1	02	1	03	0	00	1	00	00	00	9	00
12665C:	1	0	03	003	003	000	1	003	--	1	0	900	01	1	02	1	03	0	00	1	00	00	00	9	00
12895C:	1	0	03	003	003	000	1	003	--	0	0	900	01	1	02	1	03	0	00	1	00	00	00	9	00
12954C:	0	0	03	003	003	000	1	003	--	0	0	900	01	1	02	1	03	0	00	1	00	00	00	9	00
13423C:	0	0	03	003	003	000	1	003	--	1	0	900	01	1	02	1	03	0	00	1	00	00	00	9	00
13484C:	1	0	04	003	003	000	1	003	--	1	0	900	01	1	02	1	03	0	00	1	00	00	00	9	00
13586C:	1	0	04	003	003	000	1	003	--	1	0	900	01	1	02	1	03	1	00	1	00	00	00	9	00
13590C:	1	0	04	001	003	000	1	003	--	1	0	900	01	1	02	1	03	1	00	1	00	00	00	9	00
13592C:	1	0	04	000	003	000	1	003	--	1	0	900	01	1	02	1	03	1	00	1	00	00	00	9	00
13604C:	1	0	04	004	001	000	1	003	--	1	0	900	01	1	02	1	03	1	00	1	00	00	00	9	00
13606C:	1	0	04	004	000	000	1	003	--	1	0	900	01	1	02	1	03	1	00	1	00	00	00	9	00

Sat May 6 22:56:16 1989
NET: FINAL1

LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987)
SOFTWARE LOGIC SIMULATOR
SIMULATION RESULTS

PAGE: 17
COLUMN: 1

TIME	INTCLK	TRIGGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADDR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADDR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
13611C:	1	0	04	004	000	000	1	003	--	1	0	900	01	1	02	1	03	1	04	1	00	00	00	9	00
13612C:	1	0	04	004	004	000	1	003	--	1	0	900	01	1	02	1	03	1	04	1	00	00	00	9	00
13614C:	1	0	04	004	004	000	1	003	--	1	0	900	01	1	02	1	03	1	04	0	00	00	00	9	00
13667C:	1	0	04	004	004	000	1	001	--	1	0	900	01	1	02	1	03	1	04	0	00	00	00	9	00
13669C:	1	0	04	004	004	000	1	000	--	1	0	900	01	1	02	1	03	1	04	0	00	00	00	9	00
13675C:	1	0	04	004	004	000	1	004	--	1	0	900	01	1	02	1	03	1	04	0	00	00	00	9	00
13895C:	1	0	04	004	004	000	1	004	--	0	0	900	01	1	02	1	03	1	04	0	00	00	00	9	00
13954C:	0	0	04	004	004	000	1	004	--	0	0	900	01	1	02	1	03	1	04	0	00	00	00	9	00
14423C:	0	0	04	004	004	000	1	004	--	1	0	900	01	1	02	1	03	1	04	0	00	00	00	9	00
14484C:	1	0	05	004	004	000	1	004	--	1	0	900	01	1	02	1	03	1	04	0	00	00	00	9	00
14603C:	1	0	05	004	004	000	1	004	--	1	0	900	01	1	02	1	03	1	04	1	00	00	00	9	00
14608C:	1	0	05	005	004	000	1	004	--	1	0	900	01	1	02	1	03	1	04	1	00	00	00	9	00
14612C:	1	0	05	005	004	000	1	004	--	1	0	900	05	1	02	1	03	1	04	1	00	00	00	9	00
14614C:	1	0	05	005	004	000	1	004	--	1	0	901	05	1	02	1	03	1	04	1	00	00	00	9	00
14615C:	1	0	05	005	004	000	1	004	--	1	0	901	05	0	02	1	03	1	04	1	00	00	00	9	00
14616C:	1	0	05	005	005	000	1	004	--	1	0	901	05	0	02	1	03	1	04	1	00	00	00	9	00
14665C:	1	0	05	005	005	000	1	005	--	1	0	901	05	0	02	1	03	1	04	1	00	00	00	9	00
14895C:	1	0	05	005	005	000	1	005	--	0	0	901	05	0	02	1	03	1	04	1	00	00	00	9	00
14954C:	0	0	05	005	005	000	1	005	--	0	0	901	05	0	02	1	03	1	04	1	00	00	00	9	00
15423C:	0	0	05	005	005	000	1	005	--	1	0	901	05	0	02	1	03	1	04	1	00	00	00	9	00
15484C:	1	0	06	005	005	000	1	005	--	1	0	901	05	0	02	1	03	1	04	1	00	00	00	9	00
15592C:	1	0	06	004	005	000	1	005	--	1	0	901	05	1	02	1	03	1	04	1	00	00	00	9	00
15606C:	1	0	06	006	004	000	1	005	--	1	0	901	05	1	02	1	03	1	04	1	00	00	00	9	00
15609C:	1	0	06	006	004	000	1	005	--	1	0	901	05	1	06	1	03	1	04	1	00	00	00	9	00
15612C:	1	0	06	006	004	000	1	005	--	1	0	901	05	1	06	0	03	1	04	1	00	00	00	9	00
15614C:	1	0	06	006	006	000	1	007	--	1	0	901	05	1	06	0	03	1	04	1	00	00	00	9	00
15663C:	1	0	06	006	006	000	1	006	--	1	0	901	05	1	06	0	03	1	04	1	00	00	00	9	00
15669C:	1	0	06	006	006	000	1	006	--	1	0	901	05	1	06	0	03	1	04	1	00	00	00	9	00
15895C:	1	0	06	006	006	000	1	006	--	0	0	901	05	1	06	0	03	1	04	1	00	00	00	9	00
15954C:	0	0	06	006	006	000	1	006	--	0	0	901	05	1	06	0	03	1	04	1	00	00	00	9	00
16423C:	0	0	06	006	006	000	1	006	--	1	0	901	05	1	06	0	03	1	04	1	00	00	00	9	00
16484C:	1	0	07	006	006	000	1	006	--	1	0	901	05	1	06	1	03	1	04	1	00	00	00	9	00
16588C:	1	0	07	006	006	000	1	006	--	1	0	901	05	1	06	1	07	1	04	1	00	00	00	9	00
16605C:	1	0	07	006	006	000	1	006	--	1	0	901	05	1	06	1	07	0	04	1	00	00	00	9	00
16608C:	1	0	07	007	006	000	1	006	--	1	0	901	05	1	06	1	07	0	04	1	00	00	00	9	00
16616C:	1	0	07	007	007	000	1	006	--	1	0	901	05	1	06	1	07	0	04	1	00	00	00	9	00

Sat May 6 22:56:16 1989
NET: FINAL1

LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987)
SOFTWARE LOGIC SIMULATOR
SIMULATION RESULTS

TIME	INTCLK	TRIGGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
16665C:	1	0	07	007	007	000	1	007	--	1	0	901	05	1	06	1	07	0	04	1	00	00	00	9	00
16895C:	1	0	07	007	007	000	1	007	--	0	0	901	05	1	06	1	07	0	04	1	00	00	00	9	00
16954C:	0	0	07	007	007	000	1	007	--	0	0	901	05	1	06	1	07	0	04	1	00	00	00	9	00
17423C:	0	0	07	007	007	000	1	007	--	1	0	901	05	1	06	1	07	0	04	1	00	00	00	9	00
17484C:	1	0	08	007	007	000	1	007	--	1	0	901	05	1	06	1	07	0	04	1	00	00	00	9	00
17586C:	1	0	08	007	007	000	1	007	--	1	0	901	05	1	06	1	07	1	04	1	00	00	00	9	00
17588C:	1	0	08	003	007	000	1	007	--	1	0	901	05	1	06	1	07	1	04	1	00	00	00	9	00
17590C:	1	0	08	001	007	000	1	007	--	1	0	901	05	1	06	1	07	1	04	1	00	00	00	9	00
17592C:	1	0	08	000	007	000	1	007	--	1	0	901	05	1	06	1	07	1	04	1	00	00	00	9	00
17595C:	1	0	08	000	007	000	1	007	--	1	0	901	05	1	06	1	07	1	00	1	00	00	00	9	00
17602C:	1	0	08	000	003	000	1	007	--	1	0	901	05	1	06	1	07	1	00	1	00	00	00	9	00
17603C:	1	0	08	008	003	000	1	007	--	1	0	901	05	1	06	1	07	1	00	1	00	00	00	9	00
17604C:	1	0	08	008	001	000	1	007	--	1	0	901	05	1	06	1	07	1	00	1	00	00	00	9	00
17606C:	1	0	08	008	000	000	1	007	--	1	0	901	05	1	06	1	07	1	00	1	00	00	00	9	00
17611C:	1	0	08	008	008	000	1	007	--	1	0	901	05	1	06	1	07	1	08	1	00	00	00	9	00
17614C:	1	0	08	008	008	000	1	007	--	1	0	901	05	1	06	1	07	1	08	0	00	00	00	9	00
17667C:	1	0	08	008	008	000	1	005	--	1	0	901	05	1	06	1	07	1	08	0	00	00	00	9	00
17668C:	1	0	08	008	008	000	1	001	--	1	0	901	05	1	06	1	07	1	08	0	00	00	00	9	00
17669C:	1	0	08	008	008	000	1	000	--	1	0	901	05	1	06	1	07	1	08	0	00	00	00	9	00
17674C:	1	0	08	008	008	000	1	008	--	1	0	901	05	1	06	1	07	1	08	0	00	00	00	9	00
17895C:	1	0	08	008	008	000	1	008	--	0	0	901	05	1	06	1	07	1	08	0	00	00	00	9	00
17954C:	0	0	08	008	008	000	1	008	--	0	0	901	05	1	06	1	07	1	08	0	00	00	00	9	00
18423C:	0	0	08	008	008	000	1	008	--	1	0	901	05	1	06	1	07	1	08	0	00	00	00	9	00
18484C:	1	0	09	008	008	000	1	008	--	1	0	901	05	1	06	1	07	1	08	0	00	00	00	9	00
18596C:	1	0	09	008	008	000	1	008	--	1	0	901	01	1	06	1	07	1	08	0	00	00	00	9	00
18603C:	1	0	09	008	008	000	1	008	--	1	0	901	01	1	06	1	07	1	08	1	00	00	00	9	00
18608C:	1	0	09	009	008	000	1	008	--	1	0	901	01	1	06	1	07	1	08	1	00	00	00	9	00
18611C:	1	0	09	009	008	000	1	008	--	1	0	900	01	1	06	1	07	1	08	1	00	00	00	9	00
18612C:	1	0	09	009	008	000	1	008	--	1	0	900	09	1	06	1	07	1	08	1	00	00	00	9	00
18614C:	1	0	09	009	008	000	1	008	--	1	0	902	09	1	06	1	07	1	08	1	00	00	00	9	00
18615C:	1	0	09	009	008	000	1	008	--	1	0	902	09	0	06	1	07	1	08	1	00	00	00	9	00
18616C:	1	0	09	009	009	000	1	008	--	1	0	902	09	0	06	1	07	1	08	1	00	00	00	9	00
18665C:	1	0	09	009	009	000	1	009	--	1	0	902	09	0	06	1	07	1	08	1	00	00	00	9	00
18895C:	1	0	09	009	009	000	1	009	--	0	0	902	09	0	06	1	07	1	08	1	00	00	00	9	00
18954C:	0	0	09	009	009	000	1	009	--	0	0	902	09	0	06	1	07	1	08	1	00	00	00	9	00
19423C:	0	0	09	009	009	000	1	009	--	1	0	902	09	0	06	1	07	1	08	1	00	00	00	9	00

Sat May 6 22:56:16 1989
NET: FINAL1

LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987)
SOFTWARE LOGIC SIMULATOR
SIMULATION RESULTS

TIME	INTCLK	TRIGGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADDR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADDR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
19484C:	1	0	0a	009	009	000	1	009	--	1	0	902	09	0	06	1	07	1	08	1	00	00	00	9	00
19592C:	1	0	0a	008	009	000	1	009	--	1	0	902	09	1	06	1	07	1	08	1	00	00	00	9	00
19593C:	1	0	0a	008	009	000	1	009	--	1	0	902	09	1	02	1	07	1	08	1	00	00	00	9	00
19606C:	1	0	0a	00a	008	000	1	009	--	1	0	902	09	1	02	1	07	1	08	1	00	00	00	9	00
19609C:	1	0	0a	00a	008	000	1	009	--	1	0	902	09	1	0a	1	07	1	08	1	00	00	00	9	00
19612C:	1	0	0a	00a	008	000	1	009	--	1	0	902	09	1	0a	0	07	1	08	1	00	00	00	9	00
19614C:	1	0	0a	00a	00a	000	1	009	--	1	0	902	09	1	0a	0	07	1	08	1	00	00	00	9	00
19663C:	1	0	0a	00a	00a	000	1	00b	--	1	0	902	09	1	0a	0	07	1	08	1	00	00	00	9	00
19669C:	1	0	0a	00a	00a	000	1	00a	--	1	0	902	09	1	0a	0	07	1	08	1	00	00	00	9	00
19895C:	1	0	0a	00a	00a	000	1	00a	--	0	0	902	09	1	0a	0	07	1	08	1	00	00	00	9	00
19956C:	0	0	0a	00a	00a	000	1	00a	--	0	0	902	09	1	0a	0	07	1	08	1	00	00	00	9	00
20423C:	0	0	0a	00a	00a	000	1	00a	--	1	0	902	09	1	0a	0	07	1	08	1	00	00	00	9	00
20484C:	1	0	0b	00a	00a	000	1	00a	--	1	0	902	09	1	0a	0	07	1	08	1	00	00	00	9	00
20588C:	1	0	0b	00a	00a	000	1	00a	--	1	0	902	09	1	0a	1	07	1	08	1	00	00	00	9	00
20589C:	1	0	0b	00a	00a	000	1	00a	--	1	0	902	09	1	0a	1	03	1	08	1	00	00	00	9	00
20605C:	1	0	0b	00a	00a	000	1	00a	--	1	0	902	09	1	0a	1	0b	1	08	1	00	00	00	9	00
20608C:	1	0	0b	00b	00a	000	1	00a	--	1	0	902	09	1	0a	1	0b	0	08	1	00	00	00	9	00
20616C:	1	0	0b	00b	00b	000	1	00a	--	1	0	902	09	1	0a	1	0b	0	08	1	00	00	00	9	00
20665C:	1	0	0b	00b	00b	000	1	00b	--	1	0	902	09	1	0a	1	0b	0	08	1	00	00	00	9	00
20895C:	1	0	0b	00b	00b	000	1	00b	--	0	0	902	09	1	0a	1	0b	0	08	1	00	00	00	9	00
20954C:	0	0	0b	00b	00b	000	1	00b	--	0	0	902	09	1	0a	1	0b	0	08	1	00	00	00	9	00
21423C:	0	0	0b	00b	00b	000	1	00b	--	1	0	902	09	1	0a	1	0b	0	08	1	00	00	00	9	00
21484C:	1	0	0c	00b	00b	000	1	00b	--	1	0	902	09	1	0a	1	0b	0	08	1	00	00	00	9	00
21586C:	1	0	0c	00b	00b	000	1	00b	--	1	0	902	09	1	0a	1	0b	1	08	1	00	00	00	9	00
21590C:	1	0	0c	009	00b	000	1	00b	--	1	0	902	09	1	0a	1	0b	1	08	1	00	00	00	9	00
21592C:	1	0	0c	008	00b	000	1	00b	--	1	0	902	09	1	0a	1	0b	1	08	1	00	00	00	9	00
21604C:	1	0	0c	00c	009	000	1	00b	--	1	0	902	09	1	0a	1	0b	1	08	1	00	00	00	9	00
21606C:	1	0	0c	00c	008	000	1	00b	--	1	0	902	09	1	0a	1	0b	1	08	1	00	00	00	9	00
21611C:	1	0	0c	00c	008	000	1	00b	--	1	0	902	09	1	0a	1	0b	1	0c	1	00	00	00	9	00
21612C:	1	0	0c	00c	00c	000	1	00b	--	1	0	902	09	1	0a	1	0b	1	0c	1	00	00	00	9	00
21614C:	1	0	0c	00c	00c	000	1	00b	--	1	0	902	09	1	0a	1	0b	1	0c	0	00	00	00	9	00
21667C:	1	0	0c	00c	00c	000	1	009	--	1	0	902	09	1	0a	1	0b	1	0c	0	00	00	00	9	00
21669C:	1	0	0c	00c	00c	000	1	008	--	1	0	902	09	1	0a	1	0b	1	0c	0	00	00	00	9	00
21675C:	1	0	0c	00c	00c	000	1	00c	--	1	0	902	09	1	0a	1	0b	1	0c	0	00	00	00	9	00
21895C:	1	0	0c	00c	00c	000	1	00c	--	0	0	902	09	1	0a	1	0b	1	0c	0	00	00	00	9	00
21954C:	0	0	0c	00c	00c	000	1	00c	--	0	0	902	09	1	0a	1	0b	1	0c	0	00	00	00	9	00

Sat May 6 22:56:16 1989 LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987) PAGE: 20
NET: FINAL1 SOFTWARE LOGIC SIMULATOR COLUMN: 1
SIMULATION RESULTS

TIME	INTCLK	TRIGGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENADR	DPR_ADR	DPR_DATA	DPR_WE	RAM_ENADR	RAM_ADR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEW	ODD	ARROW	BLOCK	TIME_VAL
22423C:	0	0	0c	00c	00c	000	1	00c	--	1	0	902	09	1	0a	1	0b	1	0c	0	00	00	00	9	00
22484C:	1	0	0d	00c	00c	000	1	00c	--	1	0	902	09	1	0a	1	0b	1	0c	0	00	00	00	9	00
22603C:	1	0	0d	00c	00c	000	1	00c	--	1	0	902	09	1	0a	1	0b	1	0c	1	00	00	00	9	00
22608C:	1	0	0d	00d	00c	000	1	00c	--	1	0	902	09	1	0a	1	0b	1	0c	1	00	00	00	9	00
22612C:	1	0	0d	00d	00c	000	1	00c	--	1	0	902	0d	1	0a	1	0b	1	0c	1	00	00	00	9	00
22614C:	1	0	0d	00d	00c	000	1	00c	--	1	0	903	0d	1	0a	1	0b	1	0c	1	00	00	00	9	00
22615C:	1	0	0d	00d	00c	000	1	00c	--	1	0	903	0d	0	0a	1	0b	1	0c	1	00	00	00	9	00
22616C:	1	0	0d	00d	00d	000	1	00c	--	1	0	903	0d	0	0a	1	0b	1	0c	1	00	00	00	9	00
22665C:	1	0	0d	00d	00d	000	1	00d	--	1	0	903	0d	0	0a	1	0b	1	0c	1	00	00	00	9	00
22895C:	1	0	0d	00d	00d	000	1	00d	--	0	0	903	0d	0	0a	1	0b	1	0c	1	00	00	00	9	00
22954C:	0	0	0d	00d	00d	000	1	00d	--	0	0	903	0d	0	0a	1	0b	1	0c	1	00	00	00	9	00
23423C:	0	0	0d	00d	00d	000	1	00d	--	1	0	903	0d	0	0a	1	0b	1	0c	1	00	00	00	9	00
23484C:	1	0	0e	00d	00d	000	1	00d	--	1	0	903	8d	0	0a	1	0b	1	0c	1	00	00	00	9	00
23592C:	1	0	0e	00c	00d	000	1	00d	--	1	0	903	0d	1	0a	1	0b	1	0c	1	00	00	00	9	00
23606C:	1	0	0e	00e	00c	000	1	00d	--	1	0	903	0d	1	0a	1	0b	1	0c	1	00	00	00	9	00
23609C:	1	0	0e	00e	00c	000	1	00d	--	1	0	903	0d	1	0e	1	0b	1	0c	1	00	00	00	9	00
23612C:	1	0	0e	00e	00c	000	1	00d	--	1	0	903	0d	1	0e	0	0b	1	0c	1	00	00	00	9	00
23616C:	1	0	0e	00e	00e	000	1	00d	--	1	0	903	0d	1	0e	0	0b	1	0c	1	00	00	00	9	00
23663C:	1	0	0e	00e	00e	000	1	00f	--	1	0	903	0d	1	0e	0	0b	1	0c	1	00	00	00	9	00
23669C:	1	0	0e	00e	00e	000	1	00e	--	1	0	903	0d	1	0e	0	0b	1	0c	1	00	00	00	9	00
23895C:	1	0	0e	00e	00e	000	1	00e	--	0	0	903	0d	1	0e	0	0b	1	0c	1	00	00	00	9	00
23954C:	0	0	0e	00e	00e	000	1	00e	--	0	0	903	0d	1	0e	0	0b	1	0c	1	00	00	00	9	00
24423C:	0	0	0e	00e	00e	000	1	00e	--	1	0	903	0d	1	0e	0	0b	1	0c	1	00	00	00	9	00
24484C:	1	0	0f	00e	00e	000	1	00e	--	1	0	903	0d	1	0e	0	0b	1	0c	1	00	00	00	9	00
24588C:	1	0	0f	00e	00e	000	1	00e	--	1	0	903	0d	1	0e	1	0b	1	0c	1	00	00	00	9	00
24605C:	1	0	0f	00e	00e	000	1	00e	--	1	0	903	0d	1	0e	1	0f	1	0c	1	00	00	00	9	00
24608C:	1	0	0f	00f	00e	000	1	00e	--	1	0	903	0d	1	0e	1	0f	0	0c	1	00	00	00	9	00
24616C:	1	0	0f	00f	00f	000	1	00e	--	1	0	903	0d	1	0e	1	0f	0	0c	1	00	00	00	9	00
24665C:	1	0	0f	00f	00f	000	1	00f	--	1	0	903	0d	1	0e	1	0f	0	0c	1	00	00	00	9	00
24895C:	1	0	0f	00f	00f	000	1	00f	--	0	0	903	0d	1	0e	1	0f	0	0c	1	00	00	00	9	00
24954C:	0	0	0f	00f	00f	000	1	00f	--	0	0	903	0d	1	0e	1	0f	0	0c	1	00	00	00	9	00
25423C:	0	0	0f	00f	00f	000	1	00f	--	1	0	903	0d	1	0e	1	0f	0	0c	1	00	00	00	9	00
25484C:	1	0	10	00f	00f	000	1	00f	--	1	0	903	0d	1	0e	1	0f	0	0c	1	00	00	00	9	00
25586C:	1	0	10	00f	00f	000	1	00f	--	1	0	903	0d	1	0e	1	0f	1	0c	1	00	00	00	9	00
25587C:	1	0	10	007	00f	000	1	00f	--	1	0	903	0d	1	0e	1	0f	1	0c	1	00	00	00	9	00
25588C:	1	0	10	003	00f	000	1	00f	--	1	0	903	0d	1	0e	1	0f	1	0c	1	00	00	00	9	00

Sat May 6 22:56:16 1989
NET: FINAL1

LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987)
SOFTWARE LOGIC SIMULATOR
SIMULATION RESULTS

TIME	INTCLK	TRIGGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADDR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADDR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
25590C:	1	0	10	001	00f	000	1	00f	--	1	0	903	0d	1	0e	1	0f	1	0c	1	00	00	00	9	00
25592C:	1	0	10	000	00f	000	1	00f	--	1	0	903	0d	1	0e	1	0f	1	0c	1	00	00	00	9	00
25595C:	1	0	10	000	00f	000	1	00f	--	1	0	903	0d	1	0e	1	0f	1	00	1	00	00	00	9	00
25601C:	1	0	10	000	007	000	1	00f	--	1	0	903	0d	1	0e	1	0f	1	00	1	00	00	00	9	00
25602C:	1	0	10	000	003	000	1	00f	--	1	0	903	0d	1	0e	1	0f	1	00	1	00	00	00	9	00
25604C:	1	0	10	000	001	000	1	00f	--	1	0	903	0d	1	0e	1	0f	1	00	1	00	00	00	9	00
25606C:	1	0	10	000	000	000	1	00f	--	1	0	903	0d	1	0e	1	0f	1	00	1	00	00	00	9	00
25610C:	1	0	10	010	000	000	1	00f	--	1	0	903	0d	1	0e	1	0f	1	00	1	00	00	00	9	00
25611C:	1	0	10	010	000	000	1	00f	--	1	0	903	0d	1	0e	1	0f	1	10	1	00	00	00	9	00
25614C:	1	0	10	010	000	000	1	00f	--	1	0	903	0d	1	0e	1	0f	1	10	0	00	00	00	9	00
25618C:	1	0	10	010	010	000	1	00f	--	1	0	903	0d	1	0e	1	0f	1	10	0	00	00	00	9	00
25667C:	1	0	10	010	010	000	1	005	--	1	0	903	0d	1	0e	1	0f	1	10	0	00	00	00	9	00
25668C:	1	0	10	010	010	000	1	001	--	1	0	903	0d	1	0e	1	0f	1	10	0	00	00	00	9	00
25669C:	1	0	10	010	010	000	1	000	--	1	0	903	0d	1	0e	1	0f	1	10	0	00	00	00	9	00
25681C:	1	0	10	010	010	000	1	010	--	1	0	903	0d	1	0e	1	0f	1	10	0	00	00	00	9	00
25895C:	1	0	10	010	010	000	1	010	--	0	0	903	0d	1	0e	1	0f	1	10	0	00	00	00	9	00
25954C:	0	0	10	010	010	000	1	010	--	0	0	903	0d	1	0e	1	0f	1	10	0	00	00	00	9	00
26423C:	0	0	10	010	010	000	1	010	--	1	0	903	0d	1	0e	1	0f	1	10	0	00	00	00	9	00
26484C:	1	0	11	010	010	000	1	010	--	1	0	903	0d	1	0e	1	0f	1	10	0	00	00	00	9	00
26596C:	1	0	11	010	010	000	1	010	--	1	0	903	01	1	0e	1	0f	1	10	0	00	00	00	9	00
26603C:	1	0	11	010	010	000	1	010	--	1	0	903	01	1	0e	1	0f	1	10	1	00	00	00	9	00
26608C:	1	0	11	011	010	000	1	010	--	1	0	903	01	1	0e	1	0f	1	10	1	00	00	00	9	00
26611C:	1	0	11	011	010	000	1	010	--	1	0	900	01	1	0e	1	0f	1	10	1	00	00	00	9	00
26612C:	1	0	11	011	010	000	1	010	--	1	0	900	11	1	0e	1	0f	1	10	1	00	00	00	9	00
26614C:	1	0	11	011	010	000	1	010	--	1	0	904	11	1	0e	1	0f	1	10	1	00	00	00	9	00
26615C:	1	0	11	011	010	000	1	010	--	1	0	904	11	0	0e	1	0f	1	10	1	00	00	00	9	00
26616C:	1	0	11	011	011	000	1	010	--	1	0	904	11	0	0e	1	0f	1	10	1	00	00	00	9	00
26665C:	1	0	11	011	011	000	1	011	--	1	0	904	11	0	0e	1	0f	1	10	1	00	00	00	9	00
26895C:	1	0	11	011	011	000	1	011	--	0	0	904	11	0	0e	1	0f	1	10	1	00	00	00	9	00
26954C:	0	0	11	011	011	000	1	011	--	0	0	904	11	0	0e	1	0f	1	10	1	00	00	00	9	00
27423C:	0	0	11	011	011	000	1	011	--	1	0	904	11	0	0e	1	0f	1	10	1	00	00	00	9	00
27484C:	1	0	12	011	011	000	1	011	--	1	0	904	11	0	0e	1	0f	1	10	1	00	00	00	9	00
27592C:	1	0	12	010	011	000	1	011	--	1	0	904	11	1	0e	1	0f	1	10	1	00	00	00	9	00
27593C:	1	0	12	010	011	000	1	011	--	1	0	904	11	1	02	1	0f	1	10	1	00	00	00	9	00
27606C:	1	0	12	012	010	000	1	011	--	1	0	904	11	1	02	1	0f	1	10	1	00	00	00	9	00
27609C:	1	0	12	012	010	000	1	011	--	1	0	904	11	1	12	1	0f	1	10	1	00	00	00	9	00

Sat May 6 22:56:16 1989
NET: FINAL1

LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987)
SOFTWARE LOGIC SIMULATOR
SIMULATION RESULTS

TIME	INTCLK	TRIGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADR	DPR_DAT	DPR_WE	RAM_ENAB	RAM_ADR	DAT0	WE0	DAT1	WE1	DAT2	WE2	DAT3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
27612C:	1	0	12	012	010	000	1	011	--	1	0	904	11	1	12	0	0f	1	10	1	00	00	00	9	00
27614C:	1	0	12	012	012	000	1	011	--	1	0	904	11	1	12	0	0f	1	10	1	00	00	00	9	00
27663C:	1	0	12	012	012	000	1	013	--	1	0	904	11	1	12	0	0f	1	10	1	00	00	00	9	00
27669C:	1	0	12	012	012	000	1	012	--	1	0	904	11	1	12	0	0f	1	10	1	00	00	00	9	00
27895C:	1	0	12	012	012	000	1	012	--	0	0	904	11	1	12	0	0f	1	10	1	00	00	00	9	00
27954C:	0	0	12	012	012	000	1	012	--	0	0	904	11	1	12	0	0f	1	10	1	00	00	00	9	00
28423C:	0	0	12	012	012	000	1	012	--	1	0	904	11	1	12	0	0f	1	10	1	00	00	00	9	00
28484C:	1	0	13	012	012	000	1	012	--	1	0	904	11	1	12	0	0f	1	10	1	00	00	00	9	00
28588C:	1	0	13	012	012	000	1	012	--	1	0	904	11	1	12	1	0f	1	10	1	00	00	00	9	00
28589C:	1	0	13	012	012	000	1	012	--	1	0	904	11	1	12	1	03	1	10	1	00	00	00	9	00
28605C:	1	0	13	012	012	000	1	012	--	1	0	904	11	1	12	1	13	1	10	1	00	00	00	9	00
28608C:	1	0	13	013	012	000	1	012	--	1	0	904	11	1	12	1	13	0	10	1	00	00	00	9	00
28616C:	1	0	13	013	013	000	1	012	--	1	0	904	11	1	12	1	13	0	10	1	00	00	00	9	00
28665C:	1	0	13	013	013	000	1	013	--	1	0	904	11	1	12	1	13	0	10	1	00	00	00	9	00
28895C:	1	0	13	013	013	000	1	013	--	0	0	904	11	1	12	1	13	0	10	1	00	00	00	9	00
28954C:	0	0	13	013	013	000	1	013	--	0	0	904	11	1	12	1	13	0	10	1	00	00	00	9	00
29423C:	0	0	13	013	013	000	1	013	--	1	0	904	11	1	12	1	13	0	10	1	00	00	00	9	00
29484C:	1	0	14	013	013	000	1	013	--	1	0	904	11	1	12	1	13	0	10	1	00	00	00	9	00
29586C:	1	0	14	013	013	000	1	013	--	1	0	904	11	1	12	1	13	1	10	1	00	00	00	9	00
29590C:	1	0	14	011	013	000	1	013	--	1	0	904	11	1	12	1	13	1	10	1	00	00	00	9	00
29592C:	1	0	14	010	013	000	1	013	--	1	0	904	11	1	12	1	13	1	10	1	00	00	00	9	00
29604C:	1	0	14	014	011	000	1	013	--	1	0	904	11	1	12	1	13	1	10	1	00	00	00	9	00
29606C:	1	0	14	014	010	000	1	013	--	1	0	904	11	1	12	1	13	1	10	1	00	00	00	9	00
29611C:	1	0	14	014	010	000	1	013	--	1	0	904	11	1	12	1	13	1	14	1	00	00	00	9	00
29612C:	1	0	14	014	014	000	1	013	--	1	0	904	11	1	12	1	13	1	14	1	00	00	00	9	00
29614C:	1	0	14	014	014	000	1	013	--	1	0	904	11	1	12	1	13	1	14	0	00	00	00	9	00
29667C:	1	0	14	014	014	000	1	011	--	1	0	904	11	1	12	1	13	1	14	0	00	00	00	9	00
29669C:	1	0	14	014	014	000	1	010	--	1	0	904	11	1	12	1	13	1	14	0	00	00	00	9	00
29675C:	1	0	14	014	014	000	1	014	--	1	0	904	11	1	12	1	13	1	14	0	00	00	00	9	00
29895C:	1	0	14	014	014	000	1	014	--	0	0	904	11	1	12	1	13	1	14	0	00	00	00	9	00
29954C:	0	0	14	014	014	000	1	014	--	0	0	904	11	1	12	1	13	1	14	0	00	00	00	9	00
30423C:	0	0	14	014	014	000	1	014	--	1	0	904	11	1	12	1	13	1	14	0	00	00	00	9	00
30484C:	1	0	15	014	014	000	1	014	--	1	0	904	11	1	12	1	13	1	14	0	00	00	00	9	00
30603C:	1	0	15	014	014	000	1	014	--	1	0	904	11	1	12	1	13	1	14	1	00	00	00	9	00
30608C:	1	0	15	015	014	000	1	014	--	1	0	904	11	1	12	1	13	1	14	1	00	00	00	9	00
30612C:	1	0	15	015	014	000	1	014	--	1	0	904	15	1	12	1	13	1	14	1	00	00	00	9	00

Sat May 6 22:56:16 1989 LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987) PAGE: 196
NET: FINAL1 SOFTWARE LOGIC SIMULATOR COLUMN: 1
SIMULATION RESULTS

TIME	INTCLK	TRIGGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADDR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADDR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
518603C:	1	0	fd	1fc	1fc	000	1	1fc	--	1	0	97e	f9	1	fa	1	fb	1	fc	1	00	00	00	9	00
518608C:	1	0	fd	1fd	1fc	000	1	1fc	--	1	0	97e	f9	1	fa	1	fb	1	fc	1	00	00	00	9	00
518612C:	1	0	fd	1fd	1fc	000	1	1fc	--	1	0	97e	fd	1	fa	1	fb	1	fc	1	00	00	00	9	00
518614C:	1	0	fd	1fd	1fc	000	1	1fc	--	1	0	97f	fd	1	fa	1	fb	1	fc	1	00	00	00	9	00
518615C:	1	0	fd	1fd	1fc	000	1	1fc	--	1	0	97f	fd	0	fa	1	fb	1	fc	1	00	00	00	9	00
518616C:	1	0	fd	1fd	1fd	000	1	1fc	--	1	0	97f	fd	0	fa	1	fb	1	fc	1	00	00	00	9	00
518665C:	1	0	fd	1fd	1fd	000	1	1fd	--	1	0	97f	fd	0	fa	1	fb	1	fc	1	00	00	00	9	00
518895C:	1	0	fd	1fd	1fd	000	1	1fd	--	0	0	97f	fd	0	fa	1	fb	1	fc	1	00	00	00	9	00
518954C:	0	0	fd	1fd	1fd	000	1	1fd	--	0	0	97f	fd	0	fa	1	fb	1	fc	1	00	00	00	9	00
519423C:	0	0	fd	1fd	1fd	000	1	1fd	--	1	0	97f	fd	0	fa	1	fb	1	fc	1	00	00	00	9	00
519484C:	1	0	fe	1fd	1fd	000	1	1fd	--	1	0	97f	fd	0	fa	1	fb	1	fc	1	00	00	00	9	00
519592C:	1	0	fe	1fc	1fd	000	1	1fd	--	1	0	97f	fd	1	fa	1	fb	1	fc	1	00	00	00	9	00
519606C:	1	0	fe	1fe	1fc	000	1	1fd	--	1	0	97f	fd	1	fa	1	fb	1	fc	1	00	00	00	9	00
519609C:	1	0	fe	1fe	1fc	000	1	1fd	--	1	0	97f	fd	1	fe	1	fb	1	fc	1	00	00	00	9	00
519612C:	1	0	fe	1fe	1fc	000	1	1fd	--	1	0	97f	fd	1	fe	0	fb	1	fc	1	00	00	00	9	00
519614C:	1	0	fe	1fe	1fe	000	1	1fd	--	1	0	97f	fd	1	fe	0	fb	1	fc	1	00	00	00	9	00
519663C:	1	0	fe	1fe	1fe	000	1	1ff	--	1	0	97f	fd	1	fe	0	fb	1	fc	1	00	00	00	9	00
519669C:	1	0	fe	1fe	1fe	000	1	1fe	--	1	0	97f	fd	1	fe	0	fb	1	fc	1	00	00	00	9	00
519895C:	1	0	fe	1fe	1fe	000	1	1fe	--	0	0	97f	fd	1	fe	0	fb	1	fc	1	00	00	00	9	00
519954C:	0	0	fe	1fe	1fe	000	1	1fe	--	0	0	97f	fd	1	fe	0	fb	1	fc	1	00	00	00	9	00
520423C:	0	0	fe	1fe	1fe	000	1	1fe	--	1	0	97f	fd	1	fe	0	fb	1	fc	1	00	00	00	9	00
520484C:	1	0	ff	1fe	1fe	000	1	1fe	--	1	0	97f	fd	1	fe	0	fb	1	fc	1	00	00	00	9	00
520588C:	1	0	ff	1fe	1fe	000	1	1fe	--	1	0	97f	fd	1	fe	1	fb	1	fc	1	00	00	00	9	00
520605C:	1	0	ff	1fe	1fe	000	1	1fe	--	1	0	97f	fd	1	fe	1	ff	1	fc	1	00	00	00	9	00
520608C:	1	0	ff	1ff	1fe	000	1	1fe	--	1	0	97f	fd	1	fe	1	ff	0	fc	1	00	00	00	9	00
520616C:	1	0	ff	1ff	1ff	000	1	1fe	--	1	0	97f	fd	1	fe	1	ff	0	fc	1	00	00	00	9	00
520665C:	1	0	ff	1ff	1ff	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	0	fc	1	00	00	00	9	00
520895C:	1	0	ff	1ff	1ff	000	1	1ff	--	0	0	97f	fd	1	fe	1	ff	0	fc	1	00	00	00	9	00
520954C:	0	0	ff	1ff	1ff	000	1	1ff	--	0	0	97f	fd	1	fe	1	ff	0	fc	1	00	00	00	9	00
521423C:	0	0	ff	1ff	1ff	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	0	fc	1	00	00	00	9	00
521484C:	1	0	00	1ff	1ff	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	0	fc	1	00	00	00	9	00
521586C:	1	0	00	1ff	1ff	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	1	fc	1	00	00	00	9	00
521587C:	1	0	00	1f7	1ff	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	1	fc	1	00	00	00	9	00
521588C:	1	0	00	173	1ff	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	1	fc	1	00	00	00	9	00
521590C:	1	0	00	131	1ff	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	1	fc	1	00	00	00	9	00
521592C:	1	0	00	010	1ff	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	1	fc	1	00	00	00	9	00

Sat May 6 22:56:16 1989 LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987) PAGE: 197
NET: FINAL1 SOFTWARE LOGIC SIMULATOR COLUMN: 1
SIMULATION RESULTS

TIME	INTCLK	TRIGGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADDR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADDR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
521593C:	1	0	00	000	1ff	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	1	fc	1	00	00	00	9	00
521595C:	1	0	00	000	1ff	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	1	00	1	00	00	00	9	00
521601C:	1	0	00	000	1f7	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	1	00	1	00	00	00	9	00
521602C:	1	0	00	000	173	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	1	00	1	00	00	00	9	00
521604C:	1	0	00	000	031	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	1	00	1	00	00	00	9	00
521606C:	1	0	00	000	010	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	1	00	1	00	00	00	9	00
521607C:	1	0	00	000	000	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	1	00	1	00	00	00	9	00
521608C:	1	0	00	200	000	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	1	00	1	00	00	00	9	00
521612C:	1	0	00	200	200	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	1	00	1	00	00	00	9	00
521614C:	1	0	00	200	200	000	1	1ff	--	1	0	97f	fd	1	fe	1	ff	1	00	0	00	00	00	9	00
521667C:	1	0	00	200	200	000	1	1f5	--	1	0	97f	fd	1	fe	1	ff	1	00	0	00	00	00	9	00
521668C:	1	0	00	200	200	000	1	171	--	1	0	97f	fd	1	fe	1	ff	1	00	0	00	00	00	9	00
521669C:	1	0	00	200	200	000	1	170	--	1	0	97f	fd	1	fe	1	ff	1	00	0	00	00	00	9	00
521670C:	1	0	00	200	200	000	1	130	--	1	0	97f	fd	1	fe	1	ff	1	00	0	00	00	00	9	00
521672C:	1	0	00	200	200	000	1	010	--	1	0	97f	fd	1	fe	1	ff	1	00	0	00	00	00	9	00
521673C:	1	0	00	200	200	000	1	000	--	1	0	97f	fd	1	fe	1	ff	1	00	0	00	00	00	9	00
521679C:	1	0	00	200	200	000	1	200	--	1	0	97f	fd	1	fe	1	ff	1	00	0	00	00	00	9	00
521895C:	1	0	00	200	200	000	1	200	--	0	0	97f	fd	1	fe	1	ff	1	00	0	00	00	00	9	00
521954C:	0	0	00	200	200	000	1	200	--	0	0	97f	fd	1	fe	1	ff	1	00	0	00	00	00	9	00
522179C:	0	0	00	000	200	000	1	200	--	0	0	97f	fd	1	fe	1	ff	1	00	0	00	00	00	9	00
522259C:	0	0	00	000	200	000	1	000	--	0	0	97f	fd	1	fe	1	ff	1	00	0	00	00	00	9	00
522423C:	0	0	00	000	200	000	1	000	--	1	0	97f	fd	1	fe	1	ff	1	00	0	00	00	00	9	00
522484C:	1	0	01	000	200	000	1	000	--	1	0	97f	fd	1	fe	1	ff	1	00	0	00	00	00	9	00
522596C:	1	0	01	000	200	000	1	000	--	1	0	97f	01	1	fe	1	ff	1	00	0	00	00	00	9	00
522603C:	1	0	01	000	200	000	1	000	--	1	0	97f	01	1	fe	1	ff	1	00	1	00	00	00	9	00
522611C:	1	0	01	000	200	000	1	000	--	1	0	900	01	1	fe	1	ff	1	00	1	00	00	00	9	00
522615C:	1	0	01	000	200	000	1	000	--	1	0	900	01	0	fe	1	ff	1	00	1	00	00	00	9	00
522895C:	1	0	01	000	200	000	1	000	--	0	0	900	01	0	fe	1	ff	1	00	1	00	00	00	9	00
522954C:	0	0	01	000	200	000	1	000	--	0	0	900	01	0	fe	1	ff	1	00	1	00	00	00	9	00
523423C:	0	0	01	000	200	000	1	000	--	1	0	900	01	0	fe	1	ff	1	00	1	00	00	00	9	00
523484C:	1	0	02	000	200	000	1	000	--	1	0	900	01	0	fe	1	ff	1	00	1	00	00	00	9	00
523592C:	1	0	02	000	200	000	1	000	--	1	0	900	01	1	fe	1	ff	1	00	1	00	00	00	9	00
523593C:	1	0	02	000	200	000	1	000	--	1	0	900	01	1	02	1	ff	1	00	1	00	00	00	9	00
523608C:	1	0	02	001	200	000	1	000	--	1	0	900	01	1	02	1	ff	1	00	1	00	00	00	9	00
523612C:	1	0	02	001	200	000	1	000	--	1	0	900	01	1	02	0	ff	1	00	1	00	00	00	9	00
523665C:	1	0	02	001	200	000	1	001	--	1	0	900	01	1	02	0	ff	1	00	1	00	00	00	9	00

Sat May 6 22:56:16 1989
NET: FINAL1

LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987)
SOFTWARE LOGIC SIMULATOR
SIMULATION RESULTS

TIME	INTCLK	TRIGGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADDR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADDR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
523895C:	1	0	02	001	200	000	1	001	--	0	0	900	01	1	02	0	ff	1	00	1	00	00	00	9	00
523954C:	0	0	02	001	200	000	1	001	--	0	0	900	01	1	02	0	ff	1	00	1	00	00	00	9	00
524423C:	0	0	02	001	200	000	1	001	--	1	0	900	01	1	02	0	ff	1	00	1	00	00	00	9	00
524484C:	1	0	03	001	200	000	1	001	--	1	0	900	01	1	02	0	ff	1	00	1	00	00	00	9	00
524588C:	1	0	03	001	200	000	1	001	--	1	0	900	01	1	02	1	ff	1	00	1	00	00	00	9	00
524589C:	1	0	03	001	200	000	1	001	--	1	0	900	01	1	02	1	03	1	00	1	00	00	00	9	00
524592C:	1	0	03	000	200	000	1	001	--	1	0	900	01	1	02	1	03	1	00	1	00	00	00	9	00
524606C:	1	0	03	002	200	000	1	001	--	1	0	900	01	1	02	1	03	1	00	1	00	00	00	9	00
524608C:	1	0	03	002	200	000	1	001	--	1	0	900	01	1	02	1	03	0	00	1	00	00	00	9	00
524663C:	1	0	03	002	200	000	1	003	--	1	0	900	01	1	02	1	03	0	00	1	00	00	00	9	00
524669C:	1	0	03	002	200	000	1	002	--	1	0	900	01	1	02	1	03	0	00	1	00	00	00	9	00
524895C:	1	0	03	002	200	000	1	002	--	0	0	900	01	1	02	1	03	0	00	1	00	00	00	9	00
524954C:	0	0	03	002	200	000	1	002	--	0	0	900	01	1	02	1	03	0	00	1	00	00	00	9	00
525423C:	0	0	03	002	200	000	1	002	--	1	0	900	01	1	02	1	03	0	00	1	00	00	00	9	00
525484C:	1	0	04	002	200	000	1	002	--	1	0	900	01	1	02	1	03	0	00	1	00	00	00	9	00
525586C:	1	0	04	002	200	000	1	002	--	1	0	900	01	1	02	1	03	1	00	1	00	00	00	9	00
525608C:	1	0	04	003	200	000	1	002	--	1	0	900	01	1	02	1	03	1	00	1	00	00	00	9	00
525611C:	1	0	04	003	200	000	1	002	--	1	0	900	01	1	02	1	03	1	04	1	00	00	00	9	00
525614C:	1	0	04	003	200	000	1	002	--	1	0	900	01	1	02	1	03	1	04	0	00	00	00	9	00
525665C:	1	0	04	003	200	000	1	003	--	1	0	900	01	1	02	1	03	1	04	0	00	00	00	9	00
525895C:	1	0	04	003	200	000	1	003	--	0	0	900	01	1	02	1	03	1	04	0	00	00	00	9	00
525954C:	0	0	04	003	200	000	1	003	--	0	0	900	01	1	02	1	03	1	04	0	00	00	00	9	00
526423C:	0	0	04	003	200	000	1	003	--	1	0	900	01	1	02	1	03	1	04	0	00	00	00	9	00
526484C:	1	0	05	003	200	000	1	003	--	1	0	900	01	1	02	1	03	1	04	0	00	00	00	9	00
526590C:	1	0	05	001	200	000	1	003	--	1	0	900	01	1	02	1	03	1	04	0	00	00	00	9	00
526592C:	1	0	05	000	200	000	1	003	--	1	0	900	01	1	02	1	03	1	04	0	00	00	00	9	00
526603C:	1	0	05	000	200	000	1	003	--	1	0	900	01	1	02	1	03	1	04	1	00	00	00	9	00
526604C:	1	0	05	004	200	000	1	003	--	1	0	900	01	1	02	1	03	1	04	1	00	00	00	9	00
526612C:	1	0	05	004	200	000	1	003	--	1	0	900	05	1	02	1	03	1	04	1	00	00	00	9	00
526615C:	1	0	05	004	200	000	1	003	--	1	0	900	05	0	02	1	03	1	04	1	00	00	00	9	00
526667C:	1	0	05	004	200	000	1	001	--	1	0	900	05	0	02	1	03	1	04	1	00	00	00	9	00
526669C:	1	0	05	004	200	000	1	000	--	1	0	900	05	0	02	1	03	1	04	1	00	00	00	9	00
526675C:	1	0	05	004	200	000	1	004	--	1	0	900	05	0	02	1	03	1	04	1	00	00	00	9	00
526895C:	1	0	05	004	200	000	1	004	--	0	0	900	05	0	02	1	03	1	04	1	00	00	00	9	00
526954C:	0	0	05	004	200	000	1	004	--	0	0	900	05	0	02	1	03	1	04	1	00	00	00	9	00
527423C:	0	0	05	004	200	000	1	004	--	1	0	900	05	0	02	1	03	1	04	1	00	00	00	9	00

Sat May 6 22:56:16 1989 LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987) PAGE: 199
NET: FINAL1 SOFTWARE LOGIC SIMULATOR COLUMN: 1
SIMULATION RESULTS

TIME	INTCLK	TRIGGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
527484C:	1	0	06	004	200	000	1	004	--	1	0	900	05	0	02	1	03	1	04	1	00	00	00	9	00
527592C:	1	0	06	004	200	000	1	004	--	1	0	900	05	1	02	1	03	1	04	1	00	00	00	9	00
527608C:	1	0	06	005	200	000	1	004	--	1	0	900	05	1	02	1	03	1	04	1	00	00	00	9	00
527609C:	1	0	06	005	200	000	1	004	--	1	0	900	05	1	06	1	03	1	04	1	00	00	00	9	00
527612C:	1	0	06	005	200	000	1	004	--	1	0	900	05	1	06	0	03	1	04	1	00	00	00	9	00
527665C:	1	0	06	005	200	000	1	005	--	1	0	900	05	1	06	0	03	1	04	1	00	00	00	9	00
527895C:	1	0	06	005	200	000	1	005	--	0	0	900	05	1	06	0	03	1	04	1	00	00	00	9	00
527954C:	0	0	06	005	200	000	1	005	--	0	0	900	05	1	06	0	03	1	04	1	00	00	00	9	00
528423C:	0	0	06	005	200	000	1	005	--	1	0	900	05	1	06	0	03	1	04	1	00	00	00	9	00
528484C:	1	0	07	005	200	000	1	005	--	1	0	900	05	1	06	0	03	1	04	1	00	00	00	9	00
528588C:	1	0	07	005	200	000	1	005	--	1	0	900	05	1	06	1	03	1	04	1	00	00	00	9	00
528592C:	1	0	07	004	200	000	1	005	--	1	0	900	05	1	06	1	03	1	04	1	00	00	00	9	00
528605C:	1	0	07	004	200	000	1	005	--	1	0	900	05	1	06	1	07	1	04	1	00	00	00	9	00
528606C:	1	0	07	006	200	000	1	005	--	1	0	900	05	1	06	1	07	1	04	1	00	00	00	9	00
528608C:	1	0	07	006	200	000	1	005	--	1	0	900	05	1	06	1	07	0	04	1	00	00	00	9	00
528663C:	1	0	07	006	200	000	1	007	--	1	0	900	05	1	06	1	07	0	04	1	00	00	00	9	00
528669C:	1	0	07	006	200	000	1	006	--	1	0	900	05	1	06	1	07	0	04	1	00	00	00	9	00
528895C:	1	0	07	006	200	000	1	006	--	0	0	900	05	1	06	1	07	0	04	1	00	00	00	9	00
528954C:	0	0	07	006	200	000	1	006	--	0	0	900	05	1	06	1	07	0	04	1	00	00	00	9	00
529423C:	0	0	07	006	200	000	1	006	--	1	0	900	05	1	06	1	07	0	04	1	00	00	00	9	00
529484C:	1	0	08	006	200	000	1	006	--	1	0	900	05	1	06	1	07	0	04	1	00	00	00	9	00
529586C:	1	0	08	006	200	000	1	006	--	1	0	900	05	1	06	1	07	1	04	1	00	00	00	9	00
529595C:	1	0	08	006	200	000	1	006	--	1	0	900	05	1	06	1	07	1	00	1	00	00	00	9	00
529608C:	1	0	08	007	200	000	1	006	--	1	0	900	05	1	06	1	07	1	00	1	00	00	00	9	00
529611C:	1	0	08	007	200	000	1	006	--	1	0	900	05	1	06	1	07	1	08	1	00	00	00	9	00
529614C:	1	0	08	007	200	000	1	006	--	1	0	900	05	1	06	1	07	1	08	0	00	00	00	9	00
529665C:	1	0	08	007	200	000	1	007	--	1	0	900	05	1	06	1	07	1	08	0	00	00	00	9	00
529895C:	1	0	08	007	200	000	1	007	--	0	0	900	05	1	06	1	07	1	08	0	00	00	00	9	00
529954C:	0	0	08	007	200	000	1	007	--	0	0	900	05	1	06	1	07	1	08	0	00	00	00	9	00
530423C:	0	0	08	007	200	000	1	007	--	1	0	900	05	1	06	1	07	1	08	0	00	00	00	9	00
530484C:	1	0	09	007	200	000	1	007	--	1	0	900	05	1	06	1	07	1	08	0	00	00	00	9	00
530588C:	1	0	09	003	200	000	1	007	--	1	0	900	05	1	06	1	07	1	08	0	00	00	00	9	00
530590C:	1	0	09	001	200	000	1	007	--	1	0	900	05	1	06	1	07	1	08	0	00	00	00	9	00
530592C:	1	0	09	000	200	000	1	007	--	1	0	900	05	1	06	1	07	1	08	0	00	00	00	9	00
530596C:	1	0	09	000	200	000	1	007	--	1	0	900	01	1	06	1	07	1	08	0	00	00	00	9	00
530603C:	1	0	09	008	200	000	1	007	--	1	0	900	01	1	06	1	07	1	08	1	00	00	00	9	00

Sat May 6 22:56:16 1989
NET: FINAL1

LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987)
SOFTWARE LOGIC SIMULATOR
SIMULATION RESULTS

TIME	INTCLK	TRIGGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADDR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADDR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
594895C:	1	0	49	048	200	000	1	048	--	0	0	911	49	0	46	1	47	1	48	1	00	00	00	9	00
594954C:	0	0	49	048	200	000	1	048	--	0	0	911	49	0	46	1	47	1	48	1	00	00	00	9	00
595423C:	0	0	49	048	200	000	1	048	--	1	0	911	49	0	46	1	47	1	48	1	00	00	00	9	00
595484C:	1	0	4a	048	200	000	1	048	--	1	0	911	49	0	46	1	47	1	48	1	00	00	00	9	00
595592C:	1	0	4a	048	200	000	1	048	--	1	0	911	49	1	46	1	47	1	48	1	00	00	00	9	00
595593C:	1	0	4a	048	200	000	1	048	--	1	0	911	49	1	42	1	47	1	48	1	00	00	00	9	00
595608C:	1	0	4a	049	200	000	1	048	--	1	0	911	49	1	42	1	47	1	48	1	00	00	00	9	00
595609C:	1	0	4a	049	200	000	1	048	--	1	0	911	49	1	4a	1	47	1	48	1	00	00	00	9	00
595612C:	1	0	4a	049	200	000	1	048	--	1	0	911	49	1	4a	0	47	1	48	1	00	00	00	9	00
595665C:	1	0	4a	049	200	000	1	049	--	1	0	911	49	1	4a	0	47	1	48	1	00	00	00	9	00
595895C:	1	0	4a	049	200	000	1	049	--	0	0	911	49	1	4a	0	47	1	48	1	00	00	00	9	00
595954C:	0	0	4a	049	200	000	1	049	--	0	0	911	49	1	4a	0	47	1	48	1	00	00	00	9	00
596423C:	0	0	4a	049	200	000	1	049	--	1	0	911	49	1	4a	0	47	1	48	1	00	00	00	9	00
596484C:	1	0	4b	049	200	000	1	049	--	1	0	911	49	1	4a	0	47	1	48	1	00	00	00	9	00
596588C:	1	0	4b	049	200	000	1	049	--	1	0	911	49	1	4a	1	47	1	48	1	00	00	00	9	00
596589C:	1	0	4b	049	200	000	1	049	--	1	0	911	49	1	4a	1	43	1	48	1	00	00	00	9	00
596592C:	1	0	4b	048	200	000	1	049	--	1	0	911	49	1	4a	1	43	1	48	1	00	00	00	9	00
596605C:	1	0	4b	048	200	000	1	049	--	1	0	911	49	1	4a	1	4b	1	48	1	00	00	00	9	00
596606C:	1	0	4b	04a	200	000	1	049	--	1	0	911	49	1	4a	1	4b	1	48	1	00	00	00	9	00
596608C:	1	0	4b	04a	200	000	1	049	--	1	0	911	49	1	4a	1	4b	0	48	1	00	00	00	9	00
596663C:	1	0	4b	04a	200	000	1	04b	--	1	0	911	49	1	4a	1	4b	0	48	1	00	00	00	9	00
596669C:	1	0	4b	04a	200	000	1	04a	--	1	0	911	49	1	4a	1	4b	0	48	1	00	00	00	9	00
596895C:	1	0	4b	04a	200	000	1	04a	--	0	0	911	49	1	4a	1	4b	0	48	1	00	00	00	9	00
596954C:	0	0	4b	04a	200	000	1	04a	--	0	0	911	49	1	4a	1	4b	0	48	1	00	00	00	9	00
597423C:	0	0	4b	04a	200	000	1	04a	--	1	0	911	49	1	4a	1	4b	0	48	1	00	00	00	9	00
597484C:	1	0	4c	04a	200	000	1	04a	--	1	0	911	49	1	4a	1	4b	0	48	1	00	00	00	9	00
597586C:	1	0	4c	04a	200	000	1	04a	--	1	0	911	49	1	4a	1	4b	1	48	1	00	00	00	9	00
597608C:	1	0	4c	04b	200	000	1	04a	--	1	0	911	49	1	4a	1	4b	1	48	1	00	00	00	9	00
597611C:	1	0	4c	04b	200	000	1	04a	--	1	0	911	49	1	4a	1	4b	1	4c	1	00	00	00	9	00
597614C:	1	0	4c	04b	200	000	1	04a	--	1	0	911	49	1	4a	1	4b	1	4c	0	00	00	00	9	00
597665C:	1	0	4c	04b	200	000	1	04b	--	1	0	911	49	1	4a	1	4b	1	4c	0	00	00	00	9	00
597895C:	1	0	4c	04b	200	000	1	04b	--	0	0	911	49	1	4a	1	4b	1	4c	0	00	00	00	9	00
597954C:	0	0	4c	04b	200	000	1	04b	--	0	0	911	49	1	4a	1	4b	1	4c	0	00	00	00	9	00
598423C:	0	0	4c	04b	200	000	1	04b	--	1	0	911	49	1	4a	1	4b	1	4c	0	00	00	00	9	00
598484C:	1	0	4d	04b	200	000	1	04b	--	1	0	911	49	1	4a	1	4b	1	4c	0	00	00	00	9	00
598590C:	1	0	4d	049	200	000	1	04b	--	1	0	911	49	1	4a	1	4b	1	4c	0	00	00	00	9	00

Sat May 6 22:56:16 1989 LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987) PAGE: 222
NET: FINAL1 SOFTWARE LOGIC SIMULATOR COLUMN: 1
SIMULATION RESULTS

TIME	INTCLK	TRIGGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADDR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADDR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
598592C:	1	0	4d	048	200	000	1	04b	--	1	0	911	49	1	4a	1	4b	1	4c	0	00	00	00	9	00
598603C:	1	0	4d	048	200	000	1	04b	--	1	0	911	49	1	4a	1	4b	1	4c	1	00	00	00	9	00
598604C:	1	0	4d	04c	200	000	1	04b	--	1	0	911	49	1	4a	1	4b	1	4c	1	00	00	00	9	00
598611C:	1	0	4d	04c	200	000	1	04b	--	1	0	910	49	1	4a	1	4b	1	4c	1	00	00	00	9	00
598612C:	1	0	4d	04c	200	000	1	04b	--	1	0	910	4d	1	4a	1	4b	1	4c	1	00	00	00	9	00
598614C:	1	0	4d	04c	200	000	1	04b	--	1	0	912	4d	1	4a	1	4b	1	4c	1	00	00	00	9	00
598615C:	1	0	4d	04c	200	000	1	04b	--	1	0	912	4d	0	4a	1	4b	1	4c	1	00	00	00	9	00
598667C:	1	0	4d	04c	200	000	1	049	--	1	0	912	4d	0	4a	1	4b	1	4c	1	00	00	00	9	00
598669C:	1	0	4d	04c	200	000	1	048	--	1	0	912	4d	0	4a	1	4b	1	4c	1	00	00	00	9	00
598675C:	1	0	4d	04c	200	000	1	04c	--	1	0	912	4d	0	4a	1	4b	1	4c	1	00	00	00	9	00
598895C:	1	0	4d	04c	200	000	1	04c	--	0	0	912	4d	0	4a	1	4b	1	4c	1	00	00	00	9	00
598954C:	0	0	4d	04c	200	000	1	04c	--	0	0	912	4d	0	4a	1	4b	1	4c	1	00	00	00	9	00
599423C:	0	0	4d	04c	200	000	1	04c	--	1	0	912	4d	0	4a	1	4b	1	4c	1	00	00	00	9	00
599484C:	1	0	4e	04c	200	000	1	04c	--	1	0	912	4d	0	4a	1	4b	1	4c	1	00	00	00	9	00
599592C:	1	0	4e	04c	200	000	1	04c	--	1	0	912	4d	1	4a	1	4b	1	4c	1	00	00	00	9	00
599608C:	1	0	4e	04d	200	000	1	04c	--	1	0	912	4d	1	4a	1	4b	1	4c	1	00	00	00	9	00
599609C:	1	0	4e	04d	200	000	1	04c	--	1	0	912	4d	1	4e	1	4b	1	4c	1	00	00	00	9	00
599612C:	1	0	4e	04d	200	000	1	04c	--	1	0	912	4d	1	4e	0	4b	1	4c	1	00	00	00	9	00
599665C:	1	0	4e	04d	200	000	1	04d	--	1	0	912	4d	1	4e	0	4b	1	4c	1	00	00	00	9	00
599895C:	1	0	4e	04d	200	000	1	04d	--	0	0	912	4d	1	4e	0	4b	1	4c	1	00	00	00	9	00
599954C:	0	0	4e	04d	200	000	1	04d	--	0	0	912	4d	1	4e	0	4b	1	4c	1	00	00	00	9	00
600000 : TRIGGER ARRIVES HERE																									
600000C:	0	1	4e	04d	200	000	1	04d	--	0	0	912	4d	1	4e	0	4b	1	4c	1	00	00	00	9	00
600208C:	0	1	4e	04d	200	000	1	24d	--	0	0	912	4d	1	4e	0	4b	1	4c	1	00	00	00	9	00
600239C:	0	1	4e	04d	200	000	1	24c	--	0	0	912	4d	1	4e	0	4b	1	4c	1	00	00	00	9	00
600242C:	0	1	4e	04d	200	000	1	200	--	0	0	912	4d	1	4e	0	4b	1	4c	1	00	00	00	9	00
600423C:	0	1	4e	04d	200	000	1	200	--	1	0	912	4d	1	4e	0	4b	1	4c	1	00	00	00	9	00
600484C:	1	1	4f	04d	200	000	1	200	--	1	0	912	4d	1	4e	0	4b	1	4c	1	00	00	00	9	00
600588C:	1	1	4f	04d	200	000	1	200	--	1	0	912	4d	1	4e	1	4b	1	4c	1	00	00	00	9	00
600605C:	1	1	4f	04d	200	000	1	200	--	1	0	912	4d	1	4e	1	4f	1	4c	1	00	00	00	9	00
600608C:	1	1	4f	04d	200	000	1	200	--	1	0	912	4d	1	4e	1	4f	0	4c	1	00	00	00	9	00
600616C:	1	1	4f	04d	201	000	1	200	--	1	0	912	4d	1	4e	1	4f	0	4c	1	00	00	00	9	00
600673C:	1	1	4f	04d	201	000	1	201	--	1	0	912	4d	1	4e	1	4f	0	4c	1	00	00	00	9	00
600895C:	1	1	4f	04d	201	000	1	201	--	0	0	912	4d	1	4e	1	4f	0	4c	1	00	00	00	9	00

Sat May 6 22:56:16 1989
NET: FINAL1

LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987)
SOFTWARE LOGIC SIMULATOR
SIMULATION RESULTS

TIME	INTCLK	TRIGGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADDR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADDR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
600954C:	0	1	4f	04d	201	000	1	201	--	0	0	912	4d	1	4e	1	4f	0	4c	1	00	00	00	9	00
601423C:	0	1	4f	04d	201	000	1	201	--	1	0	912	4d	1	4e	1	4f	0	4c	1	00	00	00	9	00
601484C:	1	1	50	04d	201	000	1	201	--	1	0	912	4d	1	4e	1	4f	0	4c	1	00	00	00	9	00
601586C:	1	1	50	04d	201	000	1	201	--	1	0	912	4d	1	4e	1	4f	1	4c	1	00	00	00	9	00
601595C:	1	1	50	04d	201	000	1	201	--	1	0	912	4d	1	4e	1	4f	1	40	1	00	00	00	9	00
601606C:	1	1	50	04d	200	000	1	201	--	1	0	912	4d	1	4e	1	4f	1	40	1	00	00	00	9	00
601611C:	1	1	50	04d	200	000	1	201	--	1	0	912	4d	1	4e	1	4f	1	50	1	00	00	00	9	00
601614C:	1	1	50	04d	202	000	1	201	--	1	0	912	4d	1	4e	1	4f	1	50	0	00	00	00	9	00
601671C:	1	1	50	04d	202	000	1	203	--	1	0	912	4d	1	4e	1	4f	1	50	0	00	00	00	9	00
601683C:	1	1	50	04d	202	000	1	202	--	1	0	912	4d	1	4e	1	4f	1	50	0	00	00	00	9	00
601895C:	1	1	50	04d	202	000	1	202	--	0	0	912	4d	1	4e	1	4f	1	50	0	00	00	00	9	00
601954C:	0	1	50	04d	202	000	1	202	--	0	0	912	4d	1	4e	1	4f	1	50	0	00	00	00	9	00
602423C:	0	1	50	04d	202	000	1	202	--	1	0	912	4d	1	4e	1	4f	1	50	0	00	00	00	9	00
602484C:	1	1	51	04d	202	000	1	202	--	1	0	912	4d	1	4e	1	4f	1	50	0	00	00	00	9	00
602596C:	1	1	51	04d	202	000	1	202	--	1	0	912	41	1	4e	1	4f	1	50	0	00	00	00	9	00
602603C:	1	1	51	04d	202	000	1	202	--	1	0	912	41	1	4e	1	4f	1	50	1	00	00	00	9	00
602611C:	1	1	51	04d	202	000	1	202	--	1	0	900	41	1	4e	1	4f	1	50	1	00	00	00	9	00
602612C:	1	1	51	04d	202	000	1	202	--	1	0	900	51	1	4e	1	4f	1	50	1	00	00	00	9	00
602614C:	1	1	51	04d	202	000	1	202	--	1	0	980	51	1	4e	1	4f	1	50	1	00	00	00	9	00
602615C:	1	1	51	04d	202	000	1	202	--	1	0	980	51	0	4e	1	4f	1	50	1	00	00	00	9	00
602616C:	1	1	51	04d	203	000	1	202	--	1	0	980	51	0	4e	1	4f	1	50	1	00	00	00	9	00
602673C:	1	1	51	04d	203	000	1	203	--	1	0	980	51	0	4e	1	4f	1	50	1	00	00	00	9	00
602895C:	1	1	51	04d	203	000	1	203	--	0	0	980	51	0	4e	1	4f	1	50	1	00	00	00	9	00
602954C:	0	1	51	04d	203	000	1	203	--	0	0	980	51	0	4e	1	4f	1	50	1	00	00	00	9	00
603423C:	0	1	51	04d	203	000	1	203	--	1	0	980	51	0	4e	1	4f	1	50	1	00	00	00	9	00
603484C:	1	1	52	04d	203	000	1	203	--	1	0	980	51	0	4e	1	4f	1	50	1	00	00	00	9	00
603592C:	1	1	52	04d	203	000	1	203	--	1	0	980	51	1	4e	1	4f	1	50	1	00	00	00	9	00
603593C:	1	1	52	04d	203	000	1	203	--	1	0	980	51	1	42	1	4f	1	50	1	00	00	00	9	00
603604C:	1	1	52	04d	201	000	1	203	--	1	0	980	51	1	42	1	4f	1	50	1	00	00	00	9	00
603606C:	1	1	52	04d	200	000	1	203	--	1	0	980	51	1	42	1	4f	1	50	1	00	00	00	9	00
603609C:	1	1	52	04d	200	000	1	203	--	1	0	980	51	1	52	1	4f	1	50	1	00	00	00	9	00
603612C:	1	1	52	04d	204	000	1	203	--	1	0	980	51	1	52	0	4f	1	50	1	00	00	00	9	00
603681C:	1	1	52	04d	204	000	1	201	--	1	0	980	51	1	52	0	4f	1	50	1	00	00	00	9	00
603683C:	1	1	52	04d	204	000	1	204	--	1	0	980	51	1	52	0	4f	1	50	1	00	00	00	9	00
603895C:	1	1	52	04d	204	000	1	204	--	0	0	980	51	1	52	0	4f	1	50	1	00	00	00	9	00
603954C:	0	1	52	04d	204	000	1	204	--	0	0	980	51	1	52	0	4f	1	50	1	00	00	00	9	00

Sat May 6 22:56:16 1989
NET: FINAL1

LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987)
SOFTWARE LOGIC SIMULATOR
SIMULATION RESULTS

TIME	INTCLK	TRIGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADDR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADDR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
604423C:	0	1	52	04d	204	000	1	204	--	1	0	980	51	1	52	0	4f	1	50	1	00	00	00	9	00
604484C:	1	1	53	04d	204	000	1	204	--	1	0	980	51	1	52	0	4f	1	50	1	00	00	00	9	00
604588C:	1	1	53	04d	204	000	1	204	--	1	0	980	51	1	52	1	4f	1	50	1	00	00	00	9	00
604589C:	1	1	53	04d	204	000	1	204	--	1	0	980	51	1	52	1	43	1	50	1	00	00	00	9	00
604605C:	1	1	53	04d	204	000	1	204	--	1	0	980	51	1	52	1	53	1	50	1	00	00	00	9	00
604608C:	1	1	53	04d	204	000	1	204	--	1	0	980	51	1	52	1	53	0	50	1	00	00	00	9	00
604616C:	1	1	53	04d	205	000	1	204	--	1	0	980	51	1	52	1	53	0	50	1	00	00	00	9	00
604673C:	1	1	53	04d	205	000	1	205	--	1	0	980	51	1	52	1	53	0	50	1	00	00	00	9	00
604895C:	1	1	53	04d	205	000	1	205	--	0	0	980	51	1	52	1	53	0	50	1	00	00	00	9	00
604954C:	0	1	53	04d	205	000	1	205	--	0	0	980	51	1	52	1	53	0	50	1	00	00	00	9	00
605423C:	0	1	53	04d	205	000	1	205	--	1	0	980	51	1	52	1	53	0	50	1	00	00	00	9	00
605484C:	1	1	54	04d	205	000	1	205	--	1	0	980	51	1	52	1	53	0	50	1	00	00	00	9	00
605586C:	1	1	54	04d	205	000	1	205	--	1	0	980	51	1	52	1	53	1	50	1	00	00	00	9	00
605606C:	1	1	54	04d	204	000	1	205	--	1	0	980	51	1	52	1	53	1	50	1	00	00	00	9	00
605611C:	1	1	54	04d	204	000	1	205	--	1	0	980	51	1	52	1	53	1	54	1	00	00	00	9	00
605614C:	1	1	54	04d	206	000	1	205	--	1	0	980	51	1	52	1	53	1	54	0	00	00	00	9	00
605671C:	1	1	54	04d	206	000	1	207	--	1	0	980	51	1	52	1	53	1	54	0	00	00	00	9	00
605683C:	1	1	54	04d	206	000	1	206	--	1	0	980	51	1	52	1	53	1	54	0	00	00	00	9	00
605895C:	1	1	54	04d	206	000	1	206	--	0	0	980	51	1	52	1	53	1	54	0	00	00	00	9	00
605954C:	0	1	54	04d	206	000	1	206	--	0	0	980	51	1	52	1	53	1	54	0	00	00	00	9	00
606423C:	0	1	54	04d	206	000	1	206	--	1	0	980	51	1	52	1	53	1	54	0	00	00	00	9	00
606484C:	1	1	55	04d	206	000	1	206	--	1	0	980	51	1	52	1	53	1	54	0	00	00	00	9	00
606603C:	1	1	55	04d	206	000	1	206	--	1	0	980	51	1	52	1	53	1	54	1	00	00	00	9	00
606612C:	1	1	55	04d	206	000	1	206	--	1	0	980	55	1	52	1	53	1	54	1	00	00	00	9	00
606614C:	1	1	55	04d	206	000	1	206	--	1	0	981	55	1	52	1	53	1	54	1	00	00	00	9	00
606615C:	1	1	55	04d	206	000	1	206	--	1	0	981	55	0	52	1	53	1	54	1	00	00	00	9	00
606616C:	1	1	55	04d	207	000	1	206	--	1	0	981	55	0	52	1	53	1	54	1	00	00	00	9	00
606673C:	1	1	55	04d	207	000	1	207	--	1	0	981	55	0	52	1	53	1	54	1	00	00	00	9	00
606895C:	1	1	55	04d	207	000	1	207	--	0	0	981	55	0	52	1	53	1	54	1	00	00	00	9	00
606954C:	0	1	55	04d	207	000	1	207	--	0	0	981	55	0	52	1	53	1	54	1	00	00	00	9	00
607423C:	0	1	55	04d	207	000	1	207	--	1	0	981	55	0	52	1	53	1	54	1	00	00	00	9	00
607484C:	1	1	56	04d	207	000	1	207	--	1	0	981	55	0	52	1	53	1	54	1	00	00	00	9	00
607592C:	1	1	56	04d	207	000	1	207	--	1	0	981	55	1	52	1	53	1	54	1	00	00	00	9	00
607602C:	1	1	56	04d	203	000	1	207	--	1	0	981	55	1	52	1	53	1	54	1	00	00	00	9	00
607604C:	1	1	56	04d	201	000	1	207	--	1	0	981	55	1	52	1	53	1	54	1	00	00	00	9	00
607606C:	1	1	56	04d	200	000	1	207	--	1	0	981	55	1	52	1	53	1	54	1	00	00	00	9	00

Sat May 6 22:56:16 1989 LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987) PAGE: 225
NET: FINAL1 SOFTWARE LOGIC SIMULATOR COLUMN: 1
SIMULATION RESULTS

TIME	INTCLK	TRIGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADR.	DPR_DAT	DPR_WE	RAM_ENAB	RAM_ADR	DAT0	WE0	DAT1	WE1	DAT2	WE2	DAT3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
607609C:	1	1	56	04d	200	000	1	207	--	1	0	981	55	1	56	1	53	1	54	1	00	00	00	9	00
607611C:	1	1	56	04d	208	000	1	207	--	1	0	981	55	1	56	1	53	1	54	1	00	00	00	9	00
607612C:	1	1	56	04d	208	000	1	207	--	1	0	981	55	1	56	0	53	1	54	1	00	00	00	9	00
607681C:	1	1	56	04d	208	000	1	205	--	1	0	981	55	1	56	0	53	1	54	1	00	00	00	9	00
607682C:	1	1	56	04d	208	000	1	209	--	1	0	981	55	1	56	0	53	1	54	1	00	00	00	9	00
607683C:	1	1	56	04d	208	000	1	208	--	1	0	981	55	1	56	0	53	1	54	1	00	00	00	9	00
607895C:	1	1	56	04d	208	000	1	208	--	0	0	981	55	1	56	0	53	1	54	1	00	00	00	9	00
607954C:	0	1	56	04d	208	000	1	208	--	0	0	981	55	1	56	0	53	1	54	1	00	00	00	9	00
608423C:	0	1	56	04d	208	000	1	208	--	1	0	981	55	1	56	0	53	1	54	1	00	00	00	9	00
608484C:	1	1	57	04d	208	000	1	208	--	1	0	981	55	1	56	0	53	1	54	1	00	00	00	9	00
608588C:	1	1	57	04d	208	000	1	208	--	1	0	981	55	1	56	1	53	1	54	1	00	00	00	9	00
608605C:	1	1	57	04d	208	000	1	208	--	1	0	981	55	1	56	1	57	1	54	1	00	00	00	9	00
608608C:	1	1	57	04d	208	000	1	208	--	1	0	981	55	1	56	1	57	0	54	1	00	00	00	9	00
608616C:	1	1	57	04d	209	000	1	208	--	1	0	981	55	1	56	1	57	0	54	1	00	00	00	9	00
608673C:	1	1	57	04d	209	000	1	209	--	1	0	981	55	1	56	1	57	0	54	1	00	00	00	9	00
608895C:	1	1	57	04d	209	000	1	209	--	0	0	981	55	1	56	1	57	0	54	1	00	00	00	9	00
608954C:	0	1	57	04d	209	000	1	209	--	0	0	981	55	1	56	1	57	0	54	1	00	00	00	9	00
609423C:	0	1	57	04d	209	000	1	209	--	1	0	981	55	1	56	1	57	0	54	1	00	00	00	9	00
609484C:	1	1	58	04d	209	000	1	209	--	1	0	981	55	1	56	1	57	0	54	1	00	00	00	9	00
609586C:	1	1	58	04d	209	000	1	209	--	1	0	981	55	1	56	1	57	1	54	1	00	00	00	9	00
609595C:	1	1	58	04d	209	000	1	209	--	1	0	981	55	1	56	1	57	1	50	1	00	00	00	9	00
609606C:	1	1	58	04d	208	000	1	209	--	1	0	981	55	1	56	1	57	1	50	1	00	00	00	9	00
609611C:	1	1	58	04d	208	000	1	209	--	1	0	981	55	1	56	1	57	1	58	1	00	00	00	9	00
609614C:	1	1	58	04d	20a	000	1	209	--	1	0	981	55	1	56	1	57	1	58	0	00	00	00	9	00
609671C:	1	1	58	04d	20a	000	1	20b	--	1	0	981	55	1	56	1	57	1	58	0	00	00	00	9	00
609683C:	1	1	58	04d	20a	000	1	20a	--	1	0	981	55	1	56	1	57	1	58	0	00	00	00	9	00
609895C:	1	1	58	04d	20a	000	1	20a	--	0	0	981	55	1	56	1	57	1	58	0	00	00	00	9	00
609954C:	0	1	58	04d	20a	000	1	20a	--	0	0	981	55	1	56	1	57	1	58	0	00	00	00	9	00
610423C:	0	1	58	04d	20a	000	1	20a	--	1	0	981	55	1	56	1	57	1	58	0	00	00	00	9	00
610484C:	1	1	59	04d	20a	000	1	20a	--	1	0	981	55	1	56	1	57	1	58	0	00	00	00	9	00
610596C:	1	1	59	04d	20a	000	1	20a	--	1	0	981	51	1	56	1	57	1	58	0	00	00	00	9	00
610603C:	1	1	59	04d	20a	000	1	20a	--	1	0	981	51	1	56	1	57	1	58	1	00	00	00	9	00
610611C:	1	1	59	04d	20a	000	1	20a	--	1	0	980	51	1	56	1	57	1	58	1	00	00	00	9	00
610612C:	1	1	59	04d	20a	000	1	20a	--	1	0	980	59	1	56	1	57	1	58	1	00	00	00	9	00
610614C:	1	1	59	04d	20a	000	1	20a	--	1	0	982	59	1	56	1	57	1	58	1	00	00	00	9	00
610615C:	1	1	59	04d	20a	000	1	20a	--	1	0	982	59	0	56	1	57	1	58	1	00	00	00	9	00

Sat May 6 22:56:16 1989
NET: FINAL1

LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987)
SOFTWARE LOGIC SIMULATOR
SIMULATION RESULTS

TIME	INTCLK	TRIGGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADDR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADDR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
610616C:	1	1	59	04d	20b	000	1	20a	--	1	0	982	59	0	56	1	57	1	58	1	00	00	00	9	00
610673C:	1	1	59	04d	20b	000	1	20b	--	1	0	982	59	0	56	1	57	1	58	1	00	00	00	9	00
610895C:	1	1	59	04d	20b	000	1	20b	--	0	0	982	59	0	56	1	57	1	58	1	00	00	00	9	00
610954C:	0	1	59	04d	20b	000	1	20b	--	0	0	982	59	0	56	1	57	1	58	1	00	00	00	9	00
611423C:	0	1	59	04d	20b	000	1	20b	--	1	0	982	59	0	56	1	57	1	58	1	00	00	00	9	00
611484C:	1	1	5a	04d	20b	000	1	20b	--	1	0	982	59	0	56	1	57	1	58	1	00	00	00	9	00
611592C:	1	1	5a	04d	20b	000	1	20b	--	1	0	982	59	1	56	1	57	1	58	1	00	00	00	9	00
611593C:	1	1	5a	04d	20b	000	1	20b	--	1	0	982	59	1	52	1	57	1	58	1	00	00	00	9	00
611604C:	1	1	5a	04d	209	000	1	20b	--	1	0	982	59	1	52	1	57	1	58	1	00	00	00	9	00
611606C:	1	1	5a	04d	208	000	1	20b	--	1	0	982	59	1	52	1	57	1	58	1	00	00	00	9	00
611609C:	1	1	5a	04d	208	000	1	20b	--	1	0	982	59	1	5a	1	57	1	58	1	00	00	00	9	00
611612C:	1	1	5a	04d	20c	000	1	20b	--	1	0	982	59	1	5a	0	57	1	58	1	00	00	00	9	00
611681C:	1	1	5a	04d	20c	000	1	209	--	1	0	982	59	1	5a	0	57	1	58	1	00	00	00	9	00
611683C:	1	1	5a	04d	20c	000	1	20c	--	1	0	982	59	1	5a	0	57	1	58	1	00	00	00	9	00
611895C:	1	1	5a	04d	20c	000	1	20c	--	0	0	982	59	1	5a	0	57	1	58	1	00	00	00	9	00
611954C:	0	1	5a	04d	20c	000	1	20c	--	0	0	982	59	1	5a	0	57	1	58	1	00	00	00	9	00
612423C:	0	1	5a	04d	20c	000	1	20c	--	1	0	982	59	1	5a	0	57	1	58	1	00	00	00	9	00
612484C:	1	1	5b	04d	20c	000	1	20c	--	1	0	982	59	1	5a	0	57	1	58	1	00	00	00	9	00
612588C:	1	1	5b	04d	20c	000	1	20c	--	1	0	982	59	1	5a	1	57	1	58	1	00	00	00	9	00
612589C:	1	1	5b	04d	20c	000	1	20c	--	1	0	982	59	1	5a	1	53	1	58	1	00	00	00	9	00
612605C:	1	1	5b	04d	20c	000	1	20c	--	1	0	982	59	1	5a	1	5b	1	58	1	00	00	00	9	00
612608C:	1	1	5b	04d	20c	000	1	20c	--	1	0	982	59	1	5a	1	5b	0	58	1	00	00	00	9	00
612616C:	1	1	5b	04d	20d	000	1	20c	--	1	0	982	59	1	5a	1	5b	0	58	1	00	00	00	9	00
612673C:	1	1	5b	04d	20d	000	1	20d	--	1	0	982	59	1	5a	1	5b	0	58	1	00	00	00	9	00
612895C:	1	1	5b	04d	20d	000	1	20d	--	0	0	982	59	1	5a	1	5b	0	58	1	00	00	00	9	00
612954C:	0	1	5b	04d	20d	000	1	20d	--	0	0	982	59	1	5a	1	5b	0	58	1	00	00	00	9	00
613423C:	0	1	5b	04d	20d	000	1	20d	--	1	0	982	59	1	5a	1	5b	0	58	1	00	00	00	9	00
613484C:	1	1	5c	04d	20d	000	1	20d	--	1	0	982	59	1	5a	1	5b	0	58	1	00	00	00	9	00
613586C:	1	1	5c	04d	20d	000	1	20d	--	1	0	982	59	1	5a	1	5b	1	58	1	00	00	00	9	00
613606C:	1	1	5c	04d	20c	000	1	20d	--	1	0	982	59	1	5a	1	5b	1	58	1	00	00	00	9	00
613611C:	1	1	5c	04d	20c	000	1	20d	--	1	0	982	59	1	5a	1	5b	1	5c	1	00	00	00	9	00
613614C:	1	1	5c	04d	20e	000	1	20d	--	1	0	982	59	1	5a	1	5b	1	5c	0	00	00	00	9	00
613671C:	1	1	5c	04d	20e	000	1	20f	--	1	0	982	59	1	5a	1	5b	1	5c	0	00	00	00	9	00
613683C:	1	1	5c	04d	20e	000	1	20e	--	1	0	982	59	1	5a	1	5b	1	5c	0	00	00	00	9	00
613895C:	1	1	5c	04d	20e	000	1	20e	--	0	0	982	59	1	5a	1	5b	1	5c	0	00	00	00	9	00
613954C:	0	1	5c	04d	20e	000	1	20e	--	0	0	982	59	1	5a	1	5b	1	5c	0	00	00	00	9	00

Sat May 6 22:56:16 1989
NET: FINAL1

LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987)
SOFTWARE LOGIC SIMULATOR
SIMULATION RESULTS

PAGE: 381
COLUMN: 1

TIME	INTCLK	TRIGGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ENAB	DPR_ADDR	DPR_DATA	DPR_WE	RAM_ENAB	RAM_ADDR	DATA0	WE0	DATA1	WE1	DATA2	WE2	DATA3	WE3	EVEN	ODD	ARROW	BLOCK	TIME_VAL
1110616C:	1	0	4d	04d	3ff	000	1	3fe	--	1	0	9ff	4d	0	4a	1	4b	1	4c	1	00	00	00	9	00
1110673C:	1	0	4d	04d	3ff	000	1	3ff	--	1	0	9ff	4d	0	4a	1	4b	1	4c	1	00	00	00	9	00
1110866C:	1	0	4d	04d	3ff	000	1	3ff	--	1	0	9ff	4d	0	4a	1	4b	1	4c	1	00	10	00	9	00
1110895C:	1	0	4d	04d	3ff	000	1	3ff	--	0	0	9ff	4d	0	4a	1	4b	1	4c	1	00	10	00	9	00
1110954C:	0	0	4d	04d	3ff	000	1	3ff	--	0	0	9ff	4d	0	4a	1	4b	1	4c	1	00	10	00	9	00
1111423C:	0	0	4d	04d	3ff	000	1	3ff	--	1	0	9ff	4d	0	4a	1	4b	1	4c	1	00	10	00	9	00
1111486C:	1	0	4e	04d	3ff	000	1	3ff	--	1	0	9ff	4d	0	4a	1	4b	1	4c	1	00	10	00	9	00
1111592C:	1	0	4e	04d	3ff	000	1	3ff	--	1	0	9ff	4d	1	4a	1	4b	1	4c	1	00	10	00	9	00
1111609C:	1	0	4e	04d	3ff	000	1	3ff	--	1	0	9ff	4d	1	4a	1	4b	1	4c	1	00	10	00	9	00
1111612C:	1	0	4e	04d	3ff	000	1	3ff	--	1	0	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1111697C:	0	0	4e	04d	3ff	000	1	3ff	--	1	0	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1111895C:	0	0	4e	04d	3ff	000	1	3ff	--	0	0	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1112423C:	0	0	4e	04d	3ff	000	1	3ff	--	1	0	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1112895C:	0	0	4e	04d	3ff	000	1	3ff	--	0	0	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1113423C:	0	0	4e	04d	3ff	000	1	3ff	--	1	0	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1113895C:	0	0	4e	04d	3ff	000	1	3ff	--	0	0	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1114423C:	0	0	4e	04d	3ff	000	1	3ff	--	1	0	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1114857C:	0	0	4e	04d	3ff	000	1	3ff	--	1	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1114895C:	0	0	4e	04d	3ff	000	1	3ff	--	0	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1115423C:	0	0	4e	04d	3ff	000	1	3ff	--	1	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1115895C:	0	0	4e	04d	3ff	000	1	3ff	--	0	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1116423C:	0	0	4e	04d	3ff	000	1	3ff	--	1	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1116895C:	0	0	4e	04d	3ff	000	1	3ff	--	0	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1117423C:	0	0	4e	04d	3ff	000	1	3ff	--	1	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1117895C:	0	0	4e	04d	3ff	000	1	3ff	--	0	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1118423C:	0	0	4e	04d	3ff	000	1	3ff	--	1	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1118895C:	0	0	4e	04d	3ff	000	1	3ff	--	0	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1119423C:	0	0	4e	04d	3ff	000	1	3ff	--	1	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1119895C:	0	0	4e	04d	3ff	000	1	3ff	--	0	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1120423C:	0	0	4e	04d	3ff	000	1	3ff	--	1	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1120895C:	0	0	4e	04d	3ff	000	1	3ff	--	0	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1121423C:	0	0	4e	04d	3ff	000	1	3ff	--	1	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1121895C:	0	0	4e	04d	3ff	000	1	3ff	--	0	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1122423C:	0	0	4e	04d	3ff	000	1	3ff	--	1	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1122895C:	0	0	4e	04d	3ff	000	1	3ff	--	0	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00
1123423C:	0	0	4e	04d	3ff	000	1	3ff	--	1	1	9ff	4d	1	4a	0	4b	1	4c	1	00	10	00	9	00

Sat May 6 22:56:16 1989 LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987) PAGE: 492
NET: FINAL1 SOFTWARE LOGIC SIMULATOR COLUMN: 1
SIMULATION RESULTS

TIME	INTCLK	TRIGGER	SIGNAL	PRE_CNTR	POST_CNTR	MEM_CNTR	DPR_ADDR	DPR_DATA	DPR_DIR	DPR_WE	RAM_ADDR	RAM_DATA	DAT0	WE0	DAT1	WE1	DAT2	WE2	DAT3	WE3	EVEN_DRAM	ODD_DRAM	ARB_WORK	BLOCK	TIME_VAL
3103021C:	0	0	4e	04d	3ff	000	1	3ff	--	0	1	000	00	1	00	0	00	1	00	1	00	10	00	9	00
3103100C:	0	0	4e	04d	3ff	000	1	3ff	--	0	1	000	00	1	00	1	00	1	00	1	00	10	00	9	00
3103104C:	0	0	4e	04d	3ff	000	1	3ff	--	0	1	000	00	1	00	1	00	1	00	0	00	10	00	9	00
3103200C:	0	0	4e	04d	1ff	000	1	3ff	--	0	1	000	00	1	00	1	00	1	00	0	00	10	00	9	00
3103202C:	0	0	4e	04d	0ff	000	1	3ff	--	0	1	000	00	1	00	1	00	1	00	0	00	10	00	9	00
3103203C:	0	0	4e	04d	0f7	000	1	3ff	--	0	1	000	00	1	00	1	00	1	00	0	00	10	00	9	00
3103204C:	0	0	4e	04d	073	000	1	3ff	--	0	1	000	00	1	00	1	00	1	00	0	00	10	00	9	00
3103206C:	0	0	4e	04d	031	000	1	3ff	--	0	1	000	00	1	00	1	00	1	00	0	00	10	00	9	00
3103208C:	0	0	4e	04d	010	000	1	3ff	--	0	1	000	00	1	00	1	00	1	00	0	00	10	00	9	00
3103209C:	0	0	4e	04d	000	000	1	3ff	--	0	1	000	00	1	00	1	00	1	00	0	00	10	00	9	00
3103222C:	0	0	4e	045	000	000	1	3ff	--	0	1	000	00	1	00	1	00	1	00	0	00	10	00	9	00
3103223C:	0	0	4e	041	000	000	1	3ff	--	0	1	000	00	1	00	1	00	1	00	0	00	10	00	9	00
3103225C:	0	0	4e	001	000	000	1	3ff	--	0	1	000	00	1	00	1	00	1	00	0	00	10	00	9	00
3103227C:	0	0	4e	000	000	000	1	3ff	--	0	1	000	00	1	00	1	00	1	00	0	00	10	00	9	00
3103274C:	0	0	4e	000	000	000	1	3fc	--	0	1	000	00	1	00	1	00	1	00	0	00	10	00	9	00
3103277C:	0	0	4e	000	000	000	1	000	--	0	1	000	00	1	00	1	00	1	00	0	00	10	00	9	00

16105000 : SIMULATION REACHED MAXIMUM SIM TIME: NORMAL EXIT

16105000 : NETWORK STATUS SAVED UNDER NAME FINAL1.

Sat May 6 22:56:16 1989 LSI LOGIC CORPORATION (COPYRIGHT 1983, 1984, 1985, 1986, 1987) PAGE: 493
NET: FINAL1 SOFTWARE LOGIC SIMULATOR COLUMN: 1
--- Audit Trail ---

Program name: ssim
Program version: 8.1.1.2 87/11/04 14:35:05

Options and parameters (group: name=value unit):

diagnostics	: serious errors	= 0	
diagnostics	: errors	= 0	
diagnostics	: warnings	= 0	
delay condition	: sim time unit	= 0.100	ns
delay condition	: processing factor	= 0.000	
delay condition	: temperature	= 25.000	Celsius
delay condition	: voltage	= 5.000	volts
delay condition	: case	= NOM	
delay condition	: DEFLOAD	= SENTRY	
argument	: DELTYPE	= NOM	
argument	: ROMCODE	= (specified)	
argument	: PROJECTID	= paran	
argument	: CKP	= ckp1A	
argument	: TIMEBLOCK	= 1	
argument	: NOFULL	= (specified)	
argument	: NOSTATISTICS	= (specified)	
argument	: EGLITCH	= (specified)	
argument	: (1)	= FINAL1	

Files (logical, r/w, revdate, checksum, name):

FILE: FINAL1.P	r 03/08/89 14:31:00	03815946	FINAL1.PAT1A
FILE: FINAL1.S	r 05/06/89 22:48:17	02764366	FINAL1.SCL1A
FILE: FINAL1.N	w 05/06/89 22:56:16		FINAL1.NSIM1A

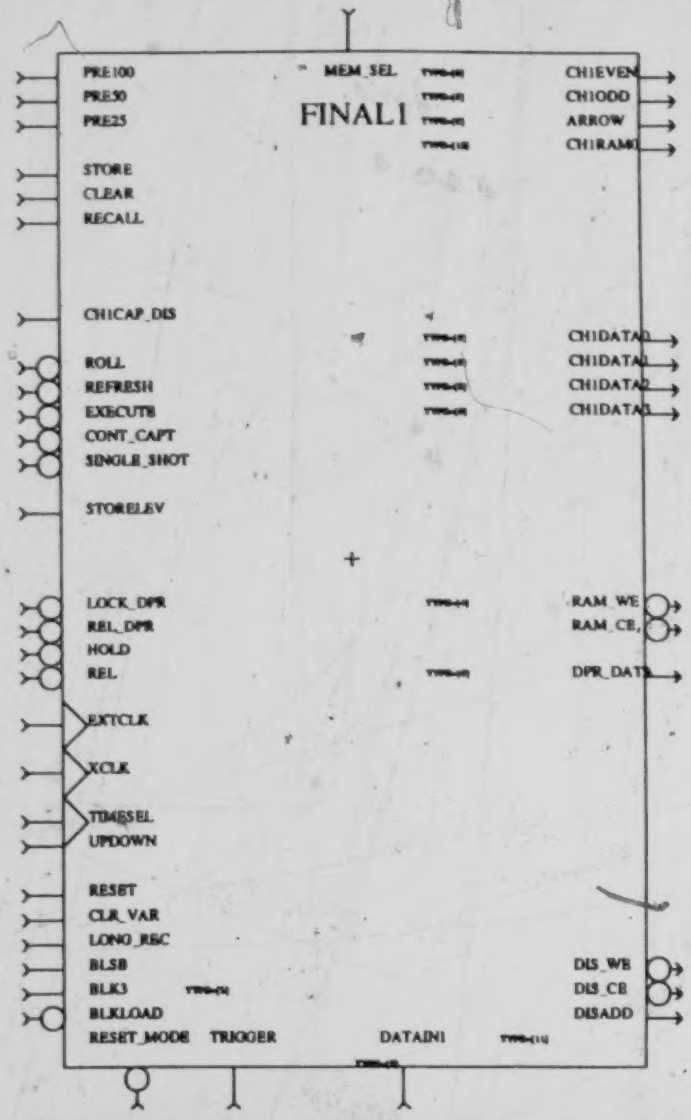
Libraries (revision, name):

LIBRARY:	revision:	name: FINAL1.NCKP1A
LIBRARY:	revision:	name: FINAL1.DSVA
LIBRARY:	revision:	name: FINAL1.SIM
LIBRARY:	revision:	name: FINAL1.NSAVA

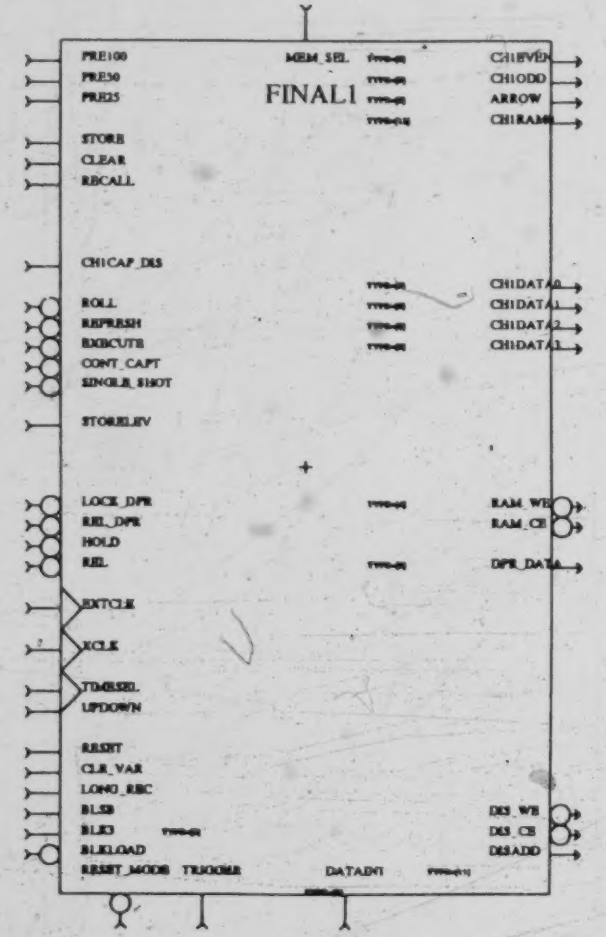
--- End of Audit Trail ---

APPENDIX B

SCHEMATIC DIAGRAMS

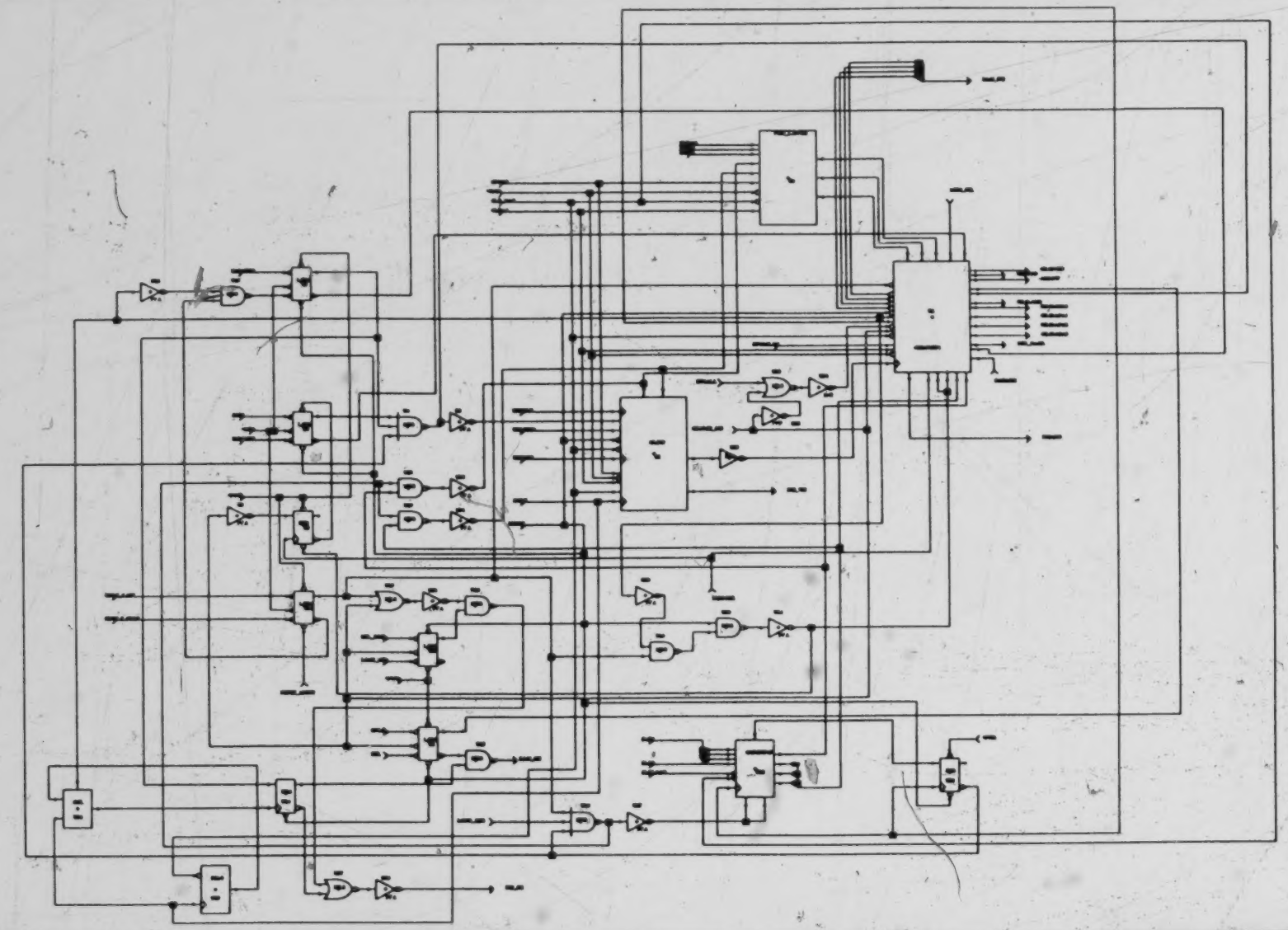


U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: DATE: MAY 9 1989 PAGE: 1 OF 63

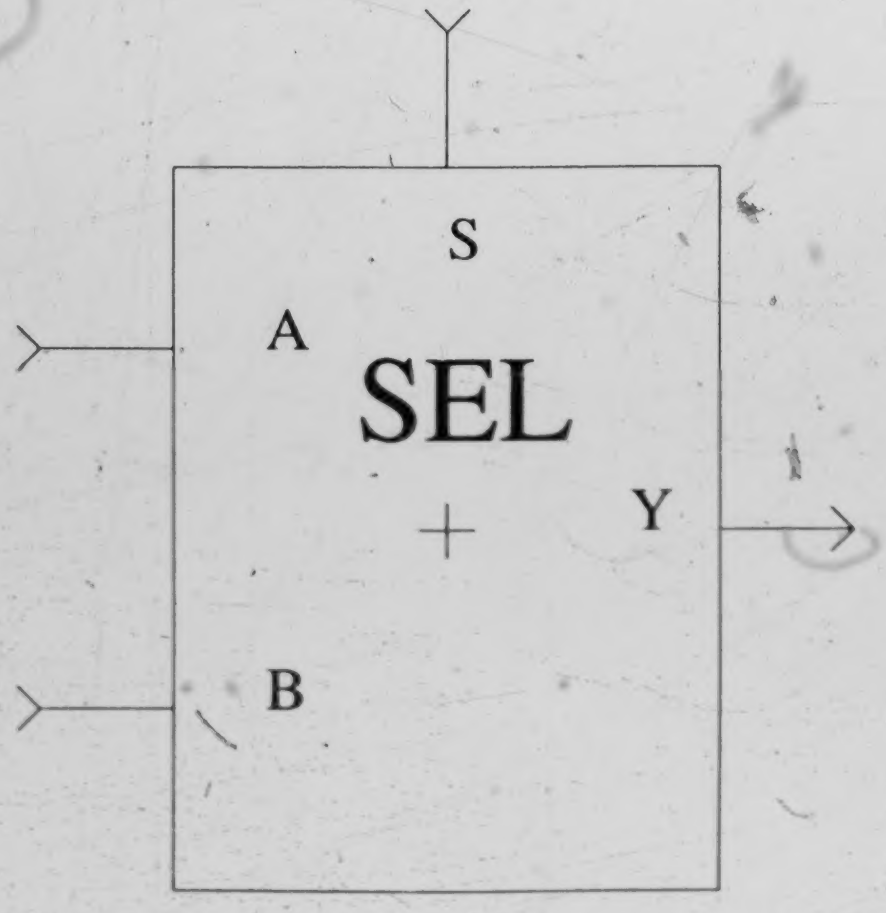


150

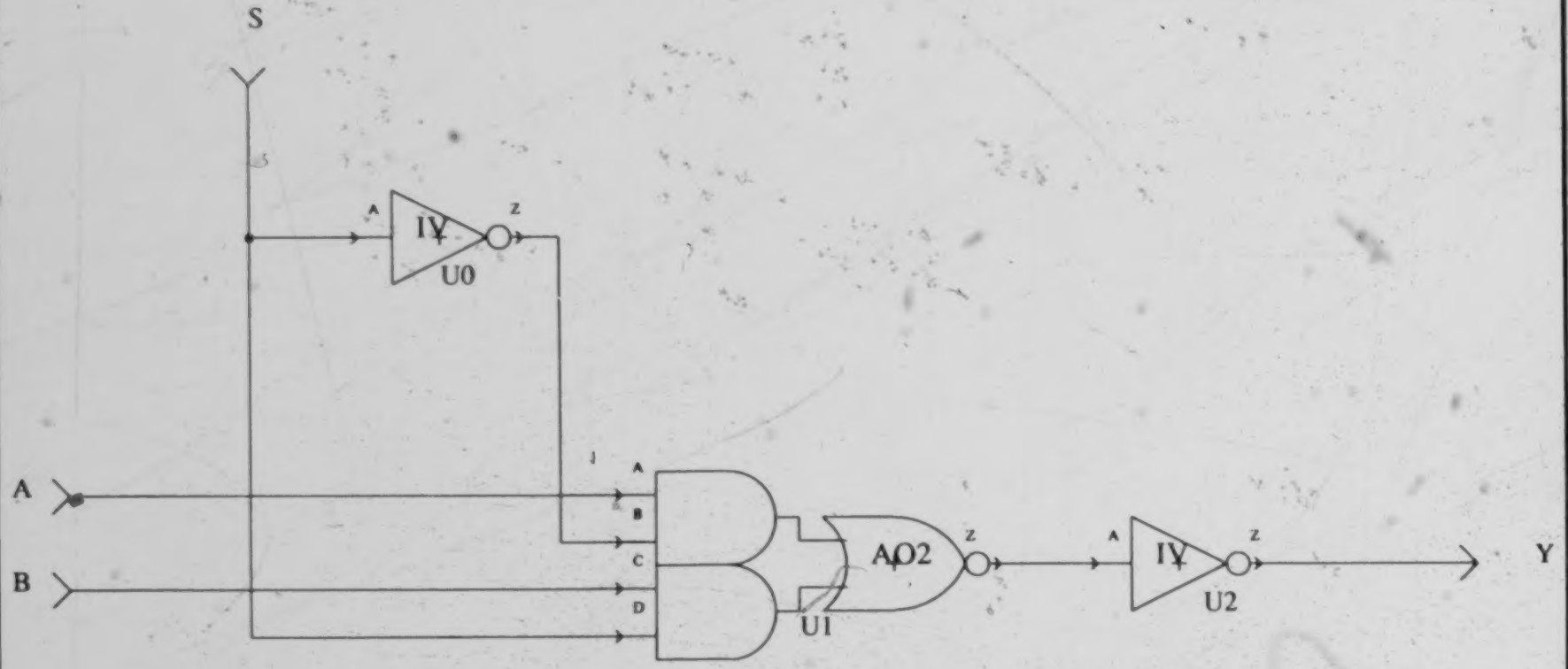
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE ARRAY DESIGN	MODULE: FINAL1 DATE: MAY 9 1989 PAGE: 2 OF 63



U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: FINAL1 DATE: MAY 9 1989 PAGE: 3 OF 63

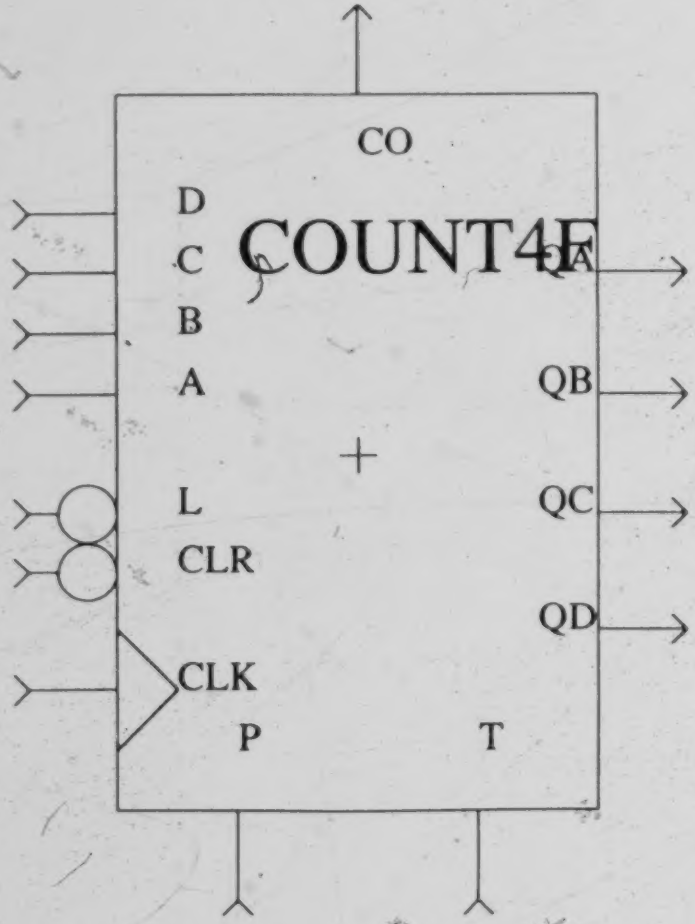


U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: SEL DATE: MAY 9 1989 PAGE: 4 OF 63

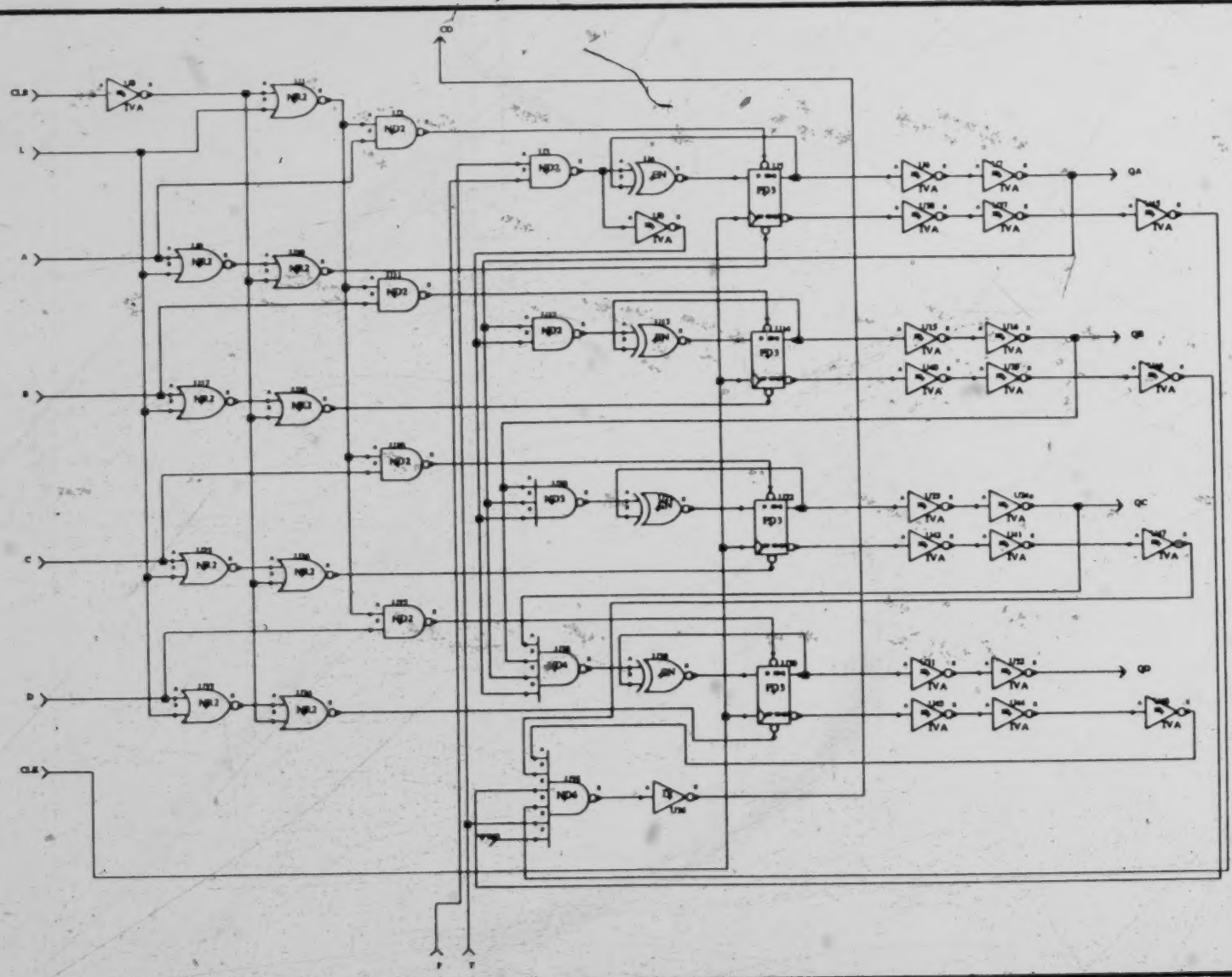


153

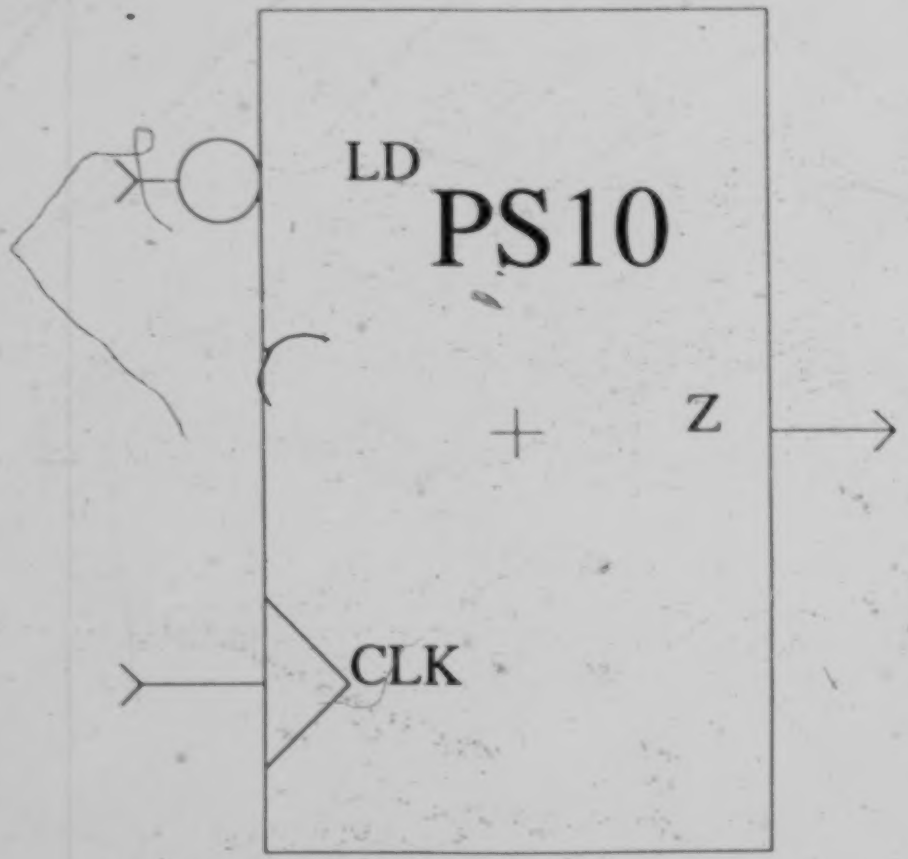
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: SEL DATE: MAY 9 1989 PAGE: 5 OF 63



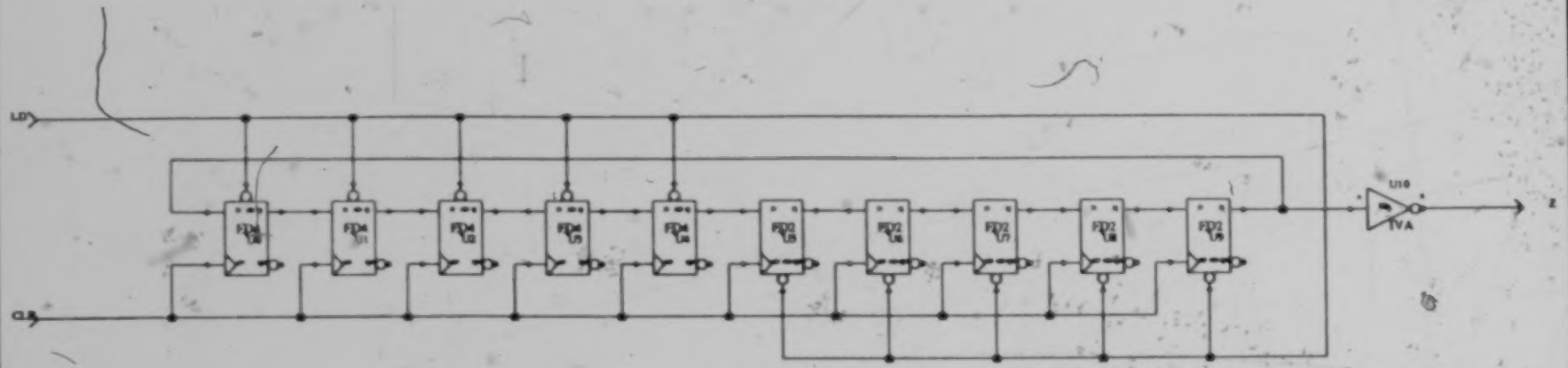
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: COUNT4P DATE: MAY 9 1989 PAGE: 6 OF 63



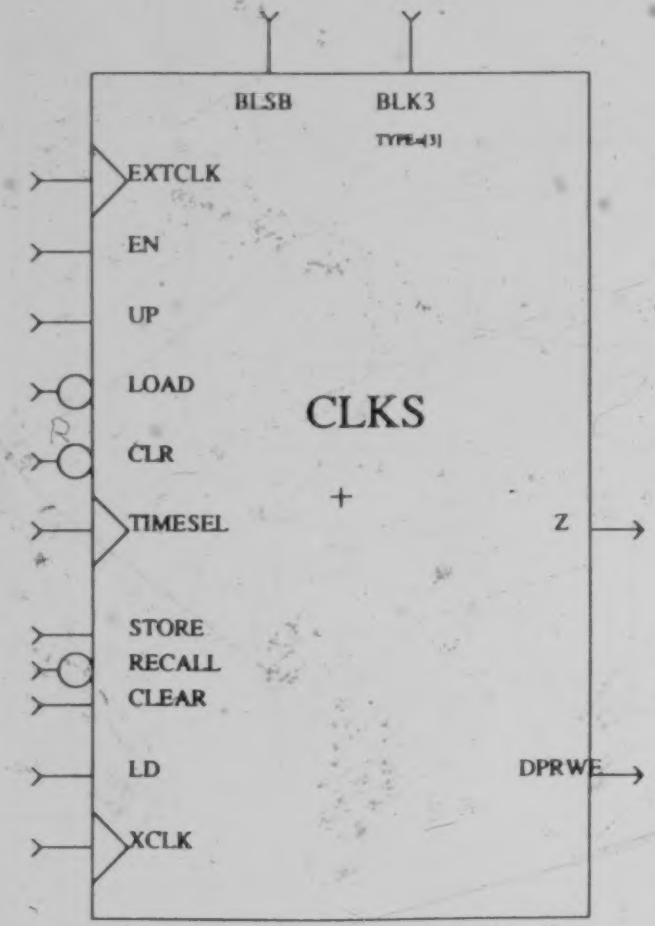
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: COUNT4F DATE: MAY 9 1989 PAGE: 7 OF 63



U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: PS10 DATE: MAY 9 1989 PAGE: 8 OF 63

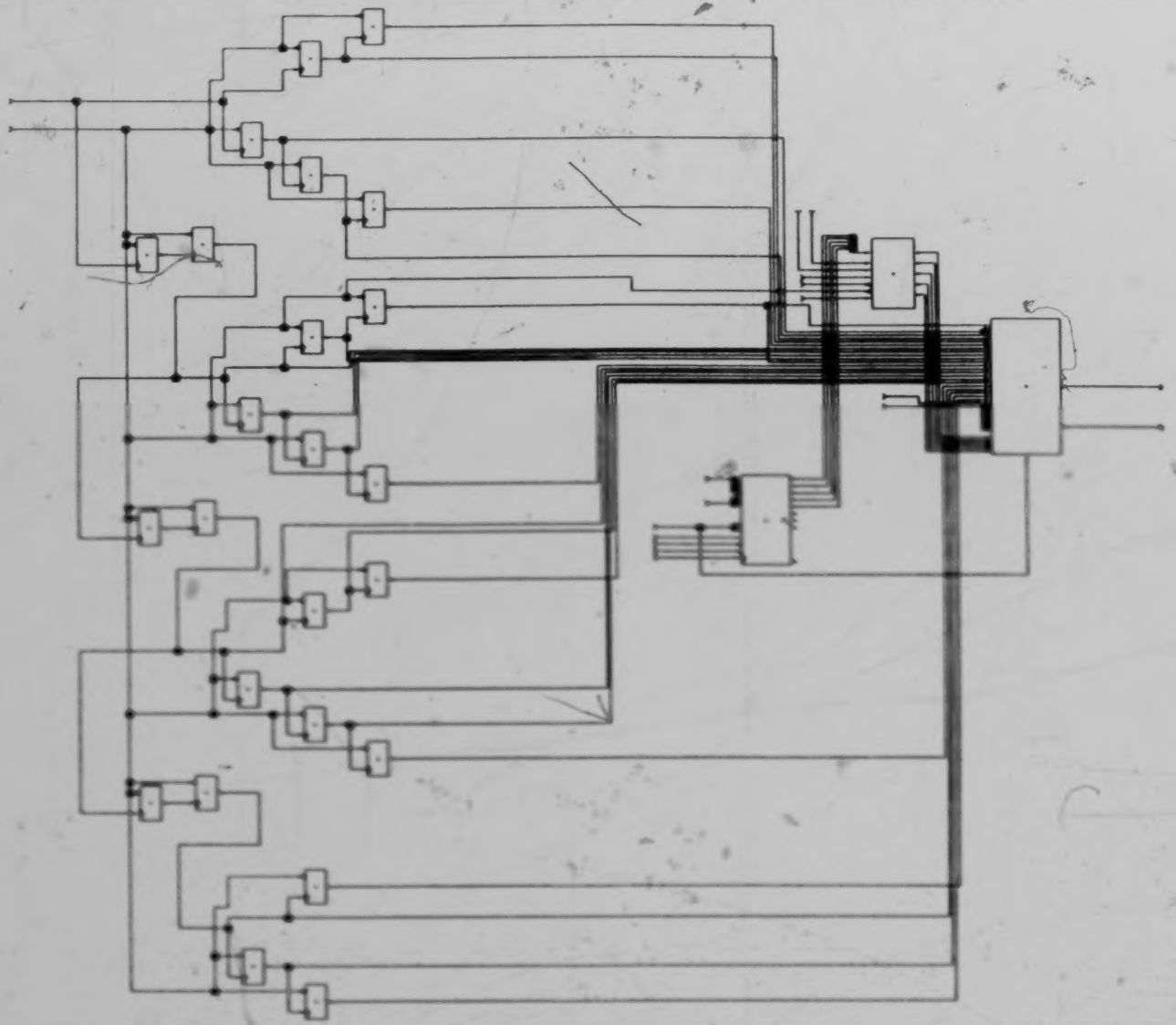


U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE ARRAY DESIGN	MODULE: PS10 DATE: MAY 9 1989 PAGE: 9 OF 63



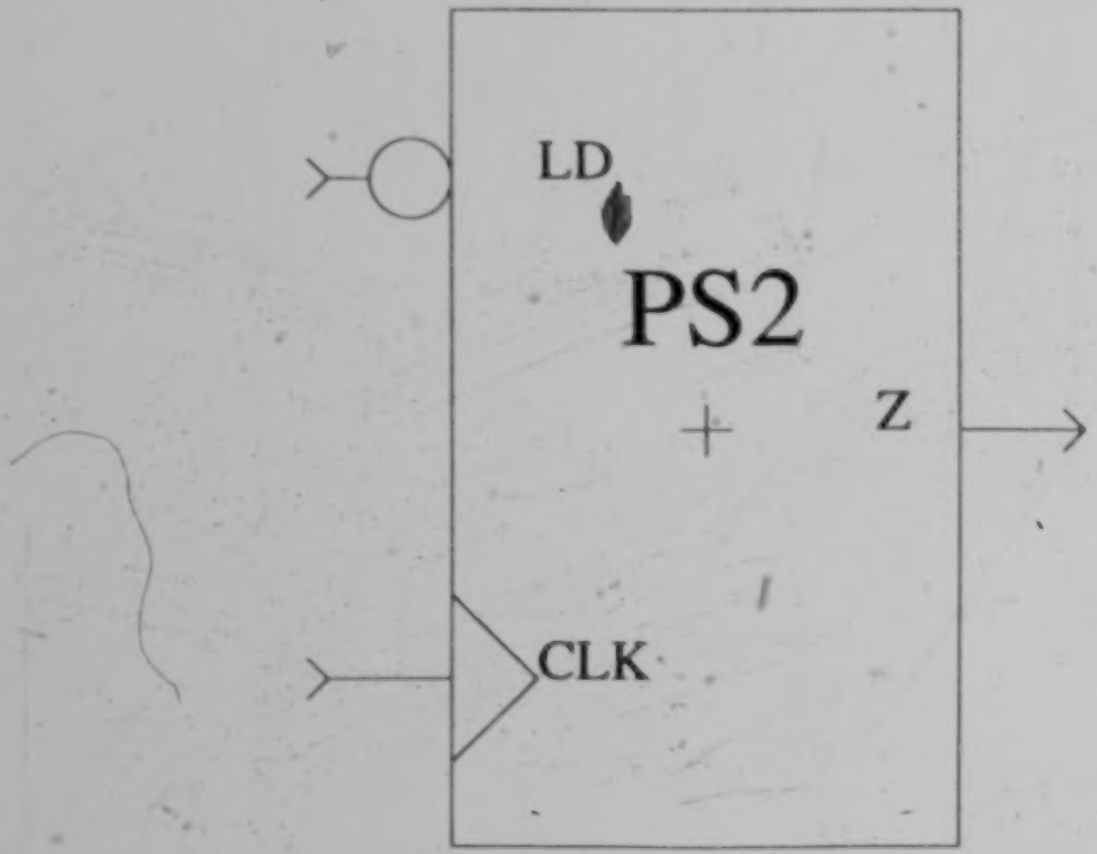
158

U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: CLKS DATE: MAY 9 1989 PAGE: 10 OF 63



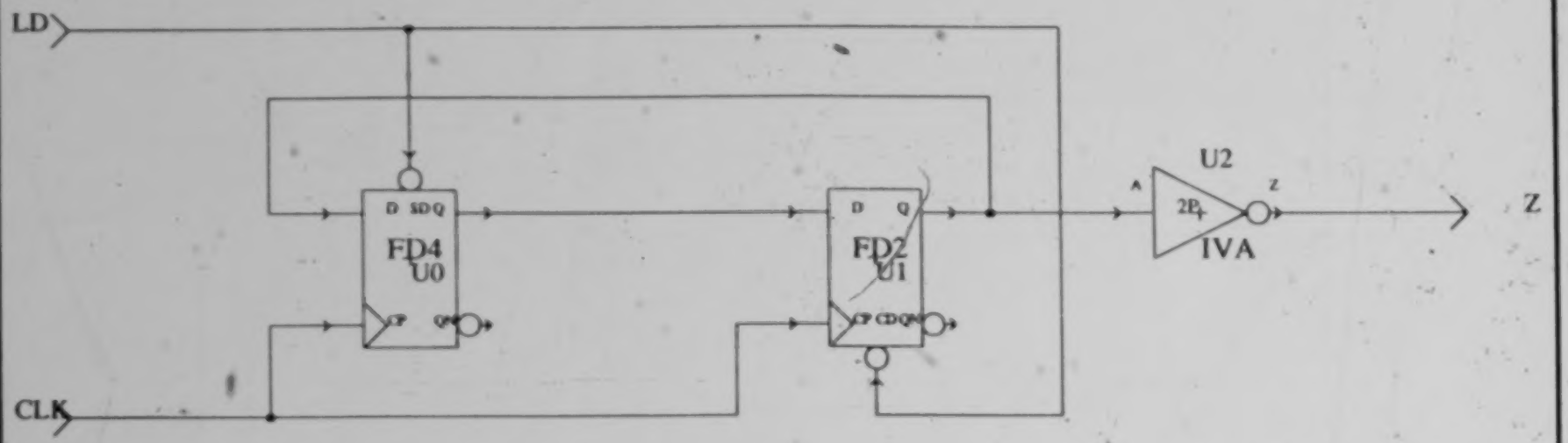
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE ARRAY DESIGN	MODULE: CLKS DATE: MAY 9 1989 PAGE: 11 OF 63

159



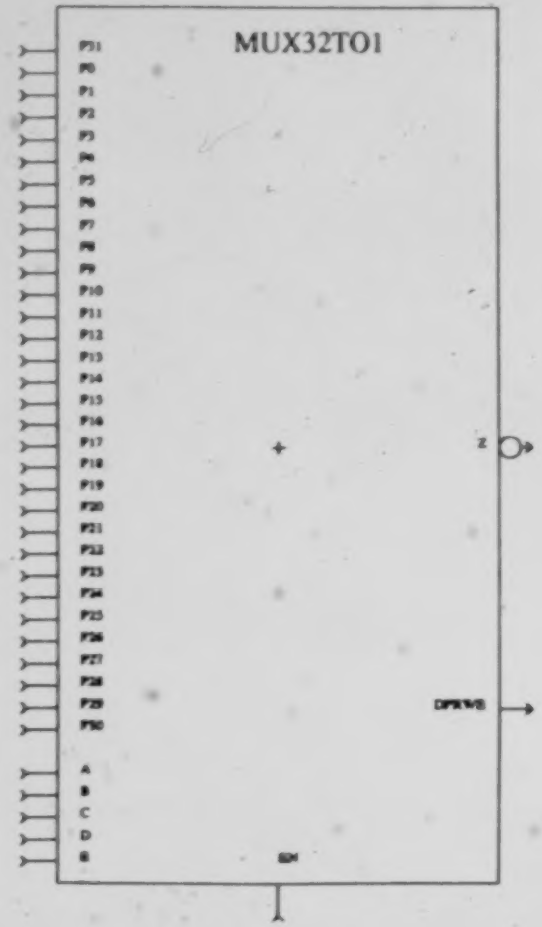
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE ARRAY DESIGN	MODULE: PS2 DATE: MAY 9 1989 PAGE: 12 OF 63

160

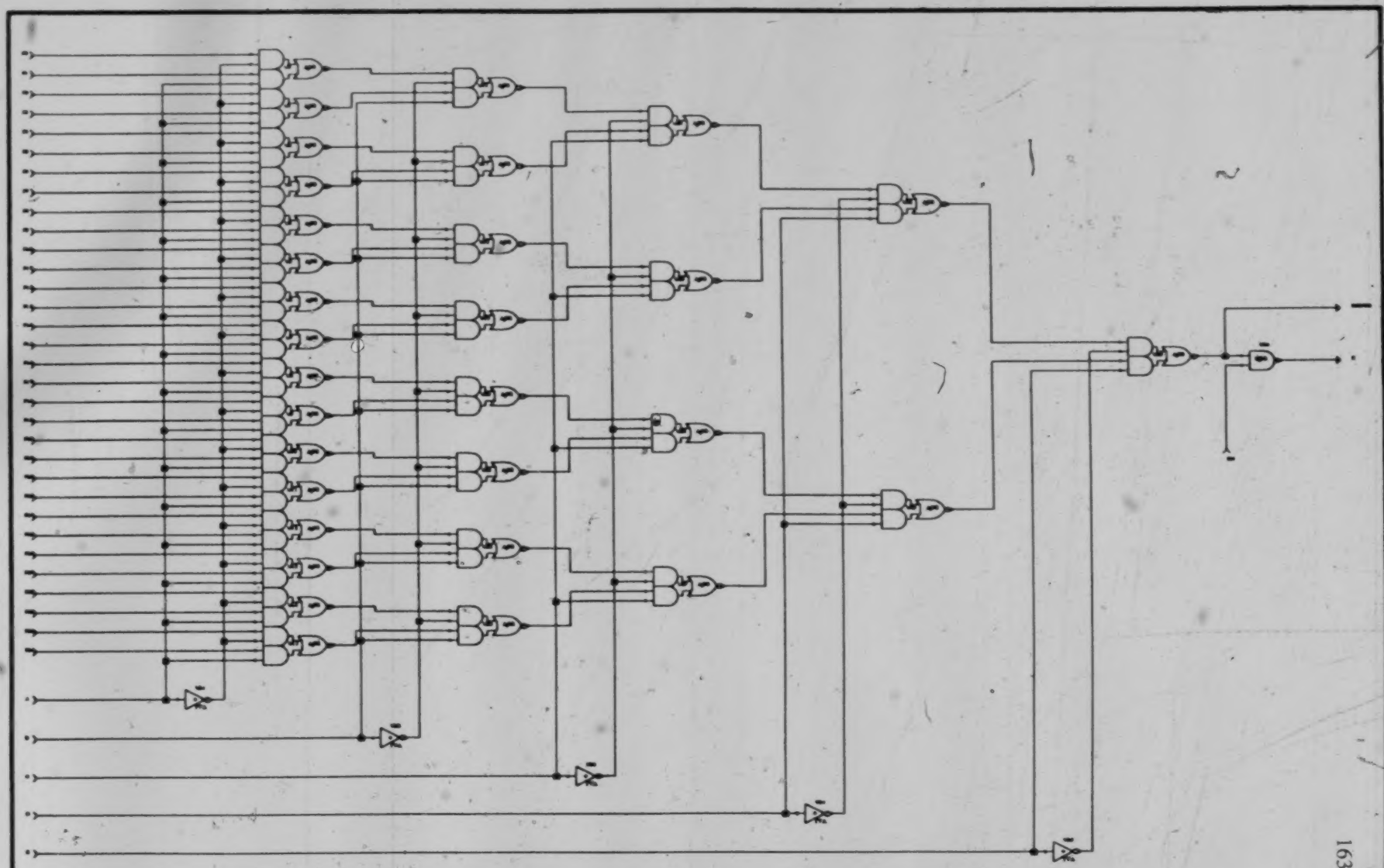


191

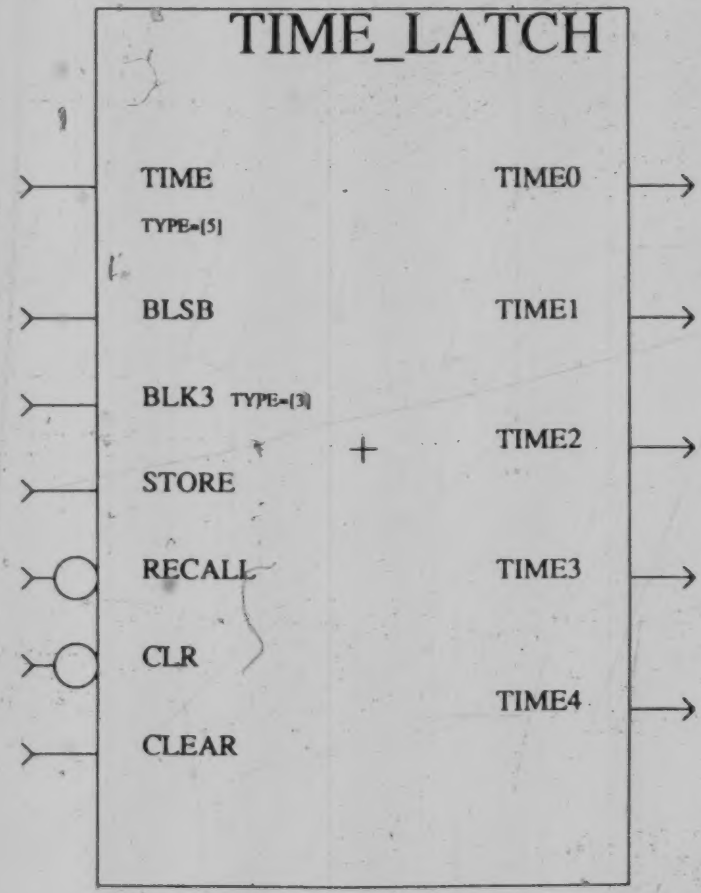
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: PS2 DATE: MAY 9 1989 PAGE: 13 OF 63



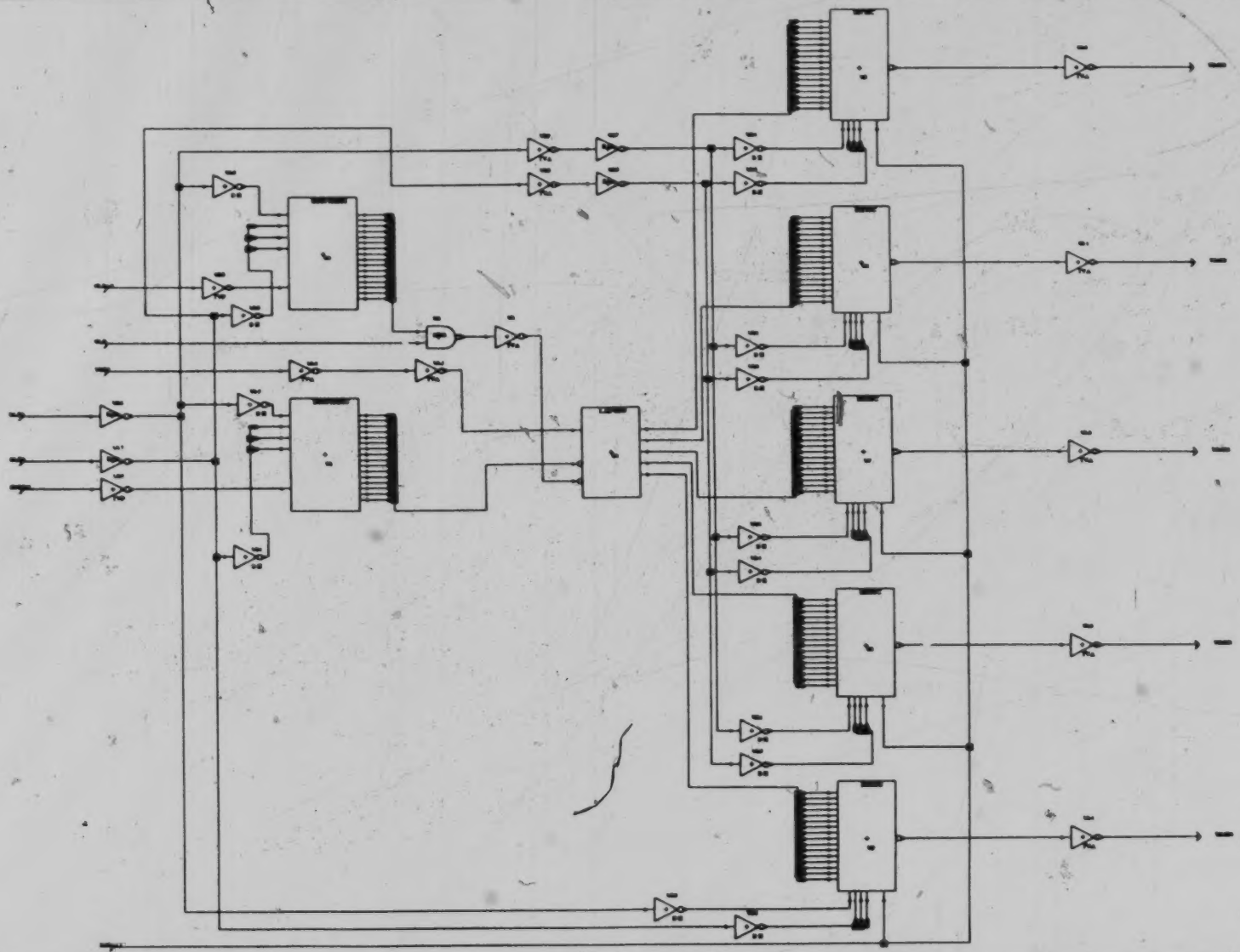
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: MUX32TO1 DATE: MAY 9 1989 PAGE: 14 OF 63



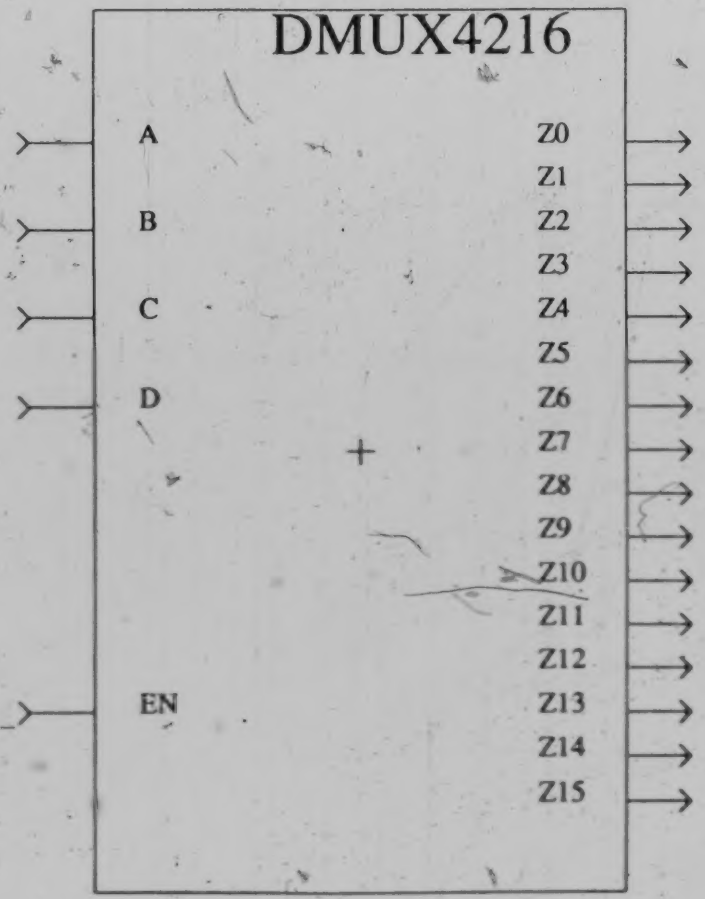
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: MUX32TO1 DATE: MAY 9 1989 PAGE: 15 OF 63



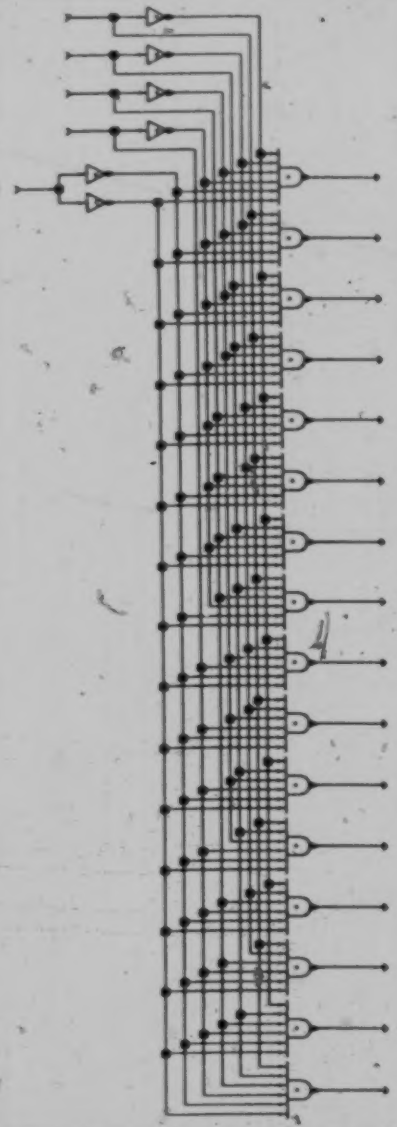
U OF A (E.E.)	DRAWN BY: M. PARANJAPE	TITLE: M.SC. PROJECT	MODULE: TIME_LAT
	CHK'D:	A DATA ACQUISITION GATE-ARRAY DESIGN	DATE: MAY 9 1989
			PAGE: 16 OF 63



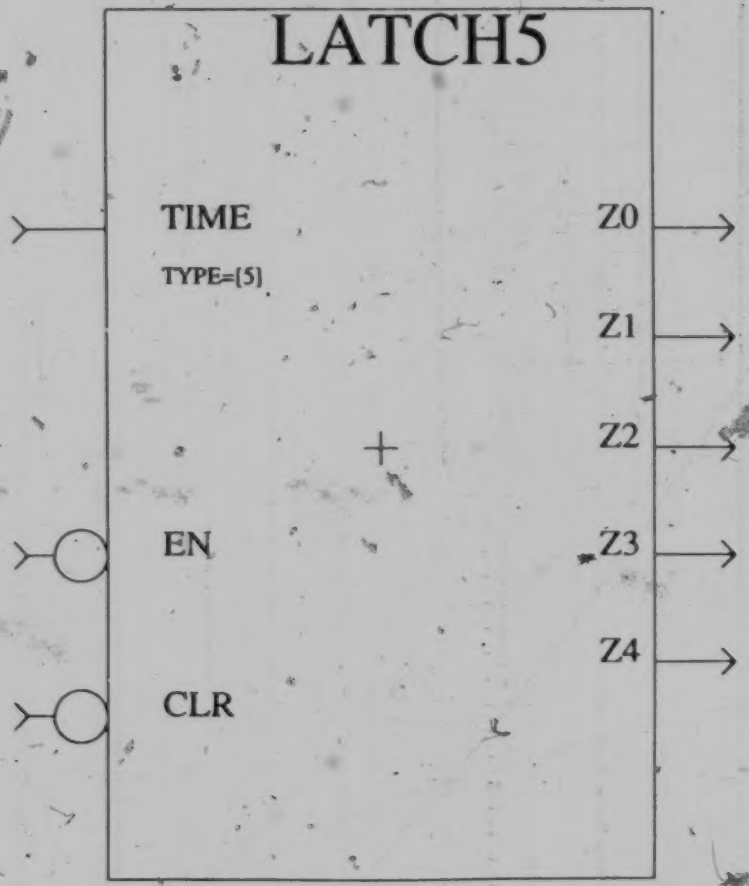
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: TIME_LAT DATE: MAY 9 1989 PAGE: 17 OF 63



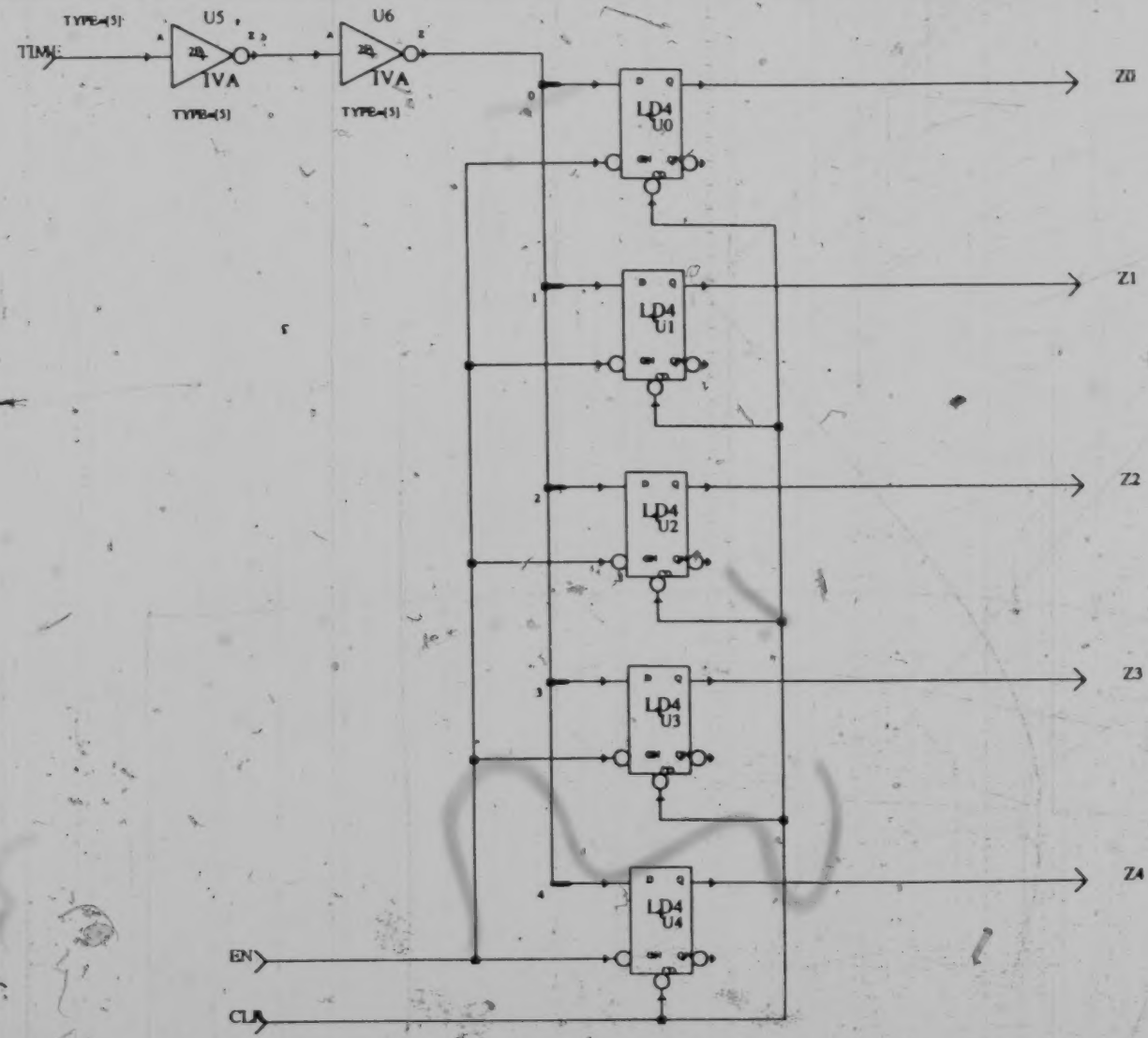
U OF A (E.E.)	DRAWN BY: M. PARANJAPE	TITLE: M.SC. PROJECT	MODULE: DMUX4216
	CHK'D:	A DATA ACQUISITION GATE-ARRAY DESIGN	DATE: MAY 9 1989
			PAGE: 18 OF 63



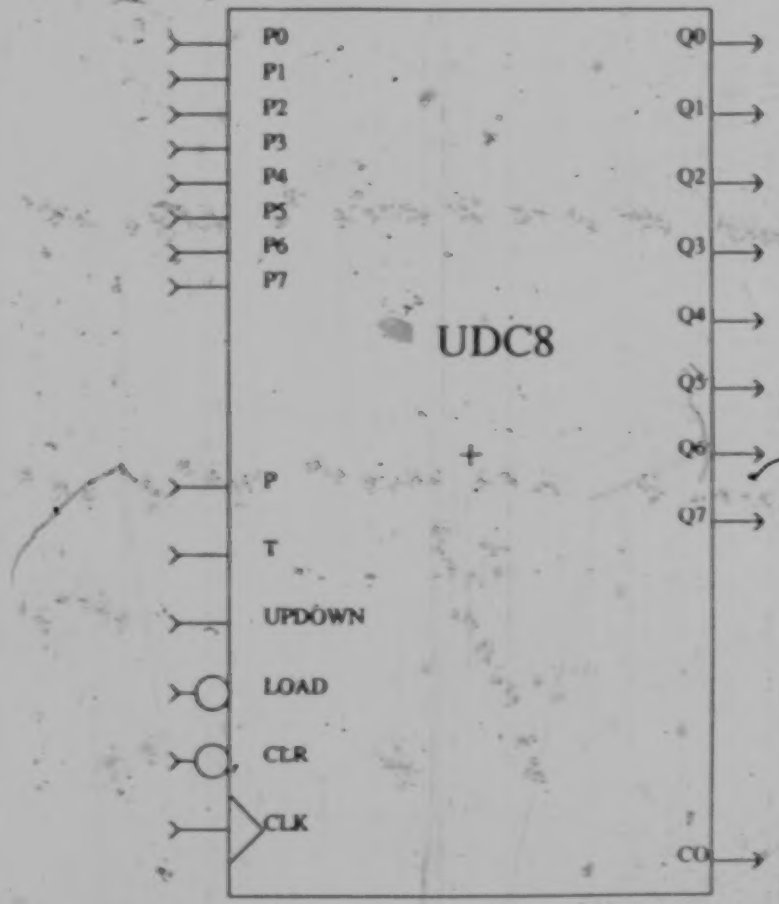
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE ARRAY DESIGN	MODULE: DMUX4216 DATE: MAY 9 1989 PAGE: 19 OF 63



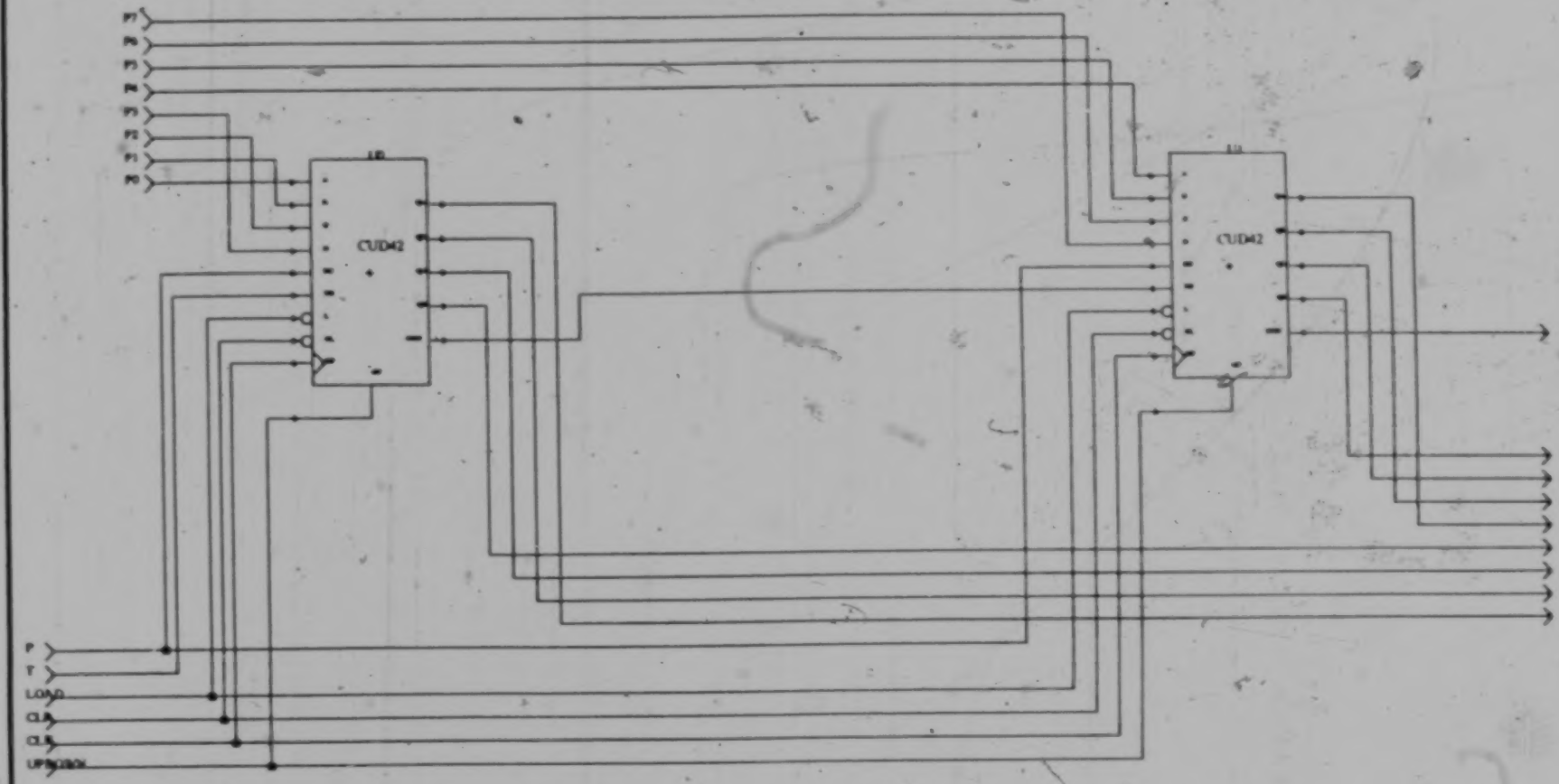
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: LATCH5 DATE: MAY 9 1989 PAGE: 20 OF 63



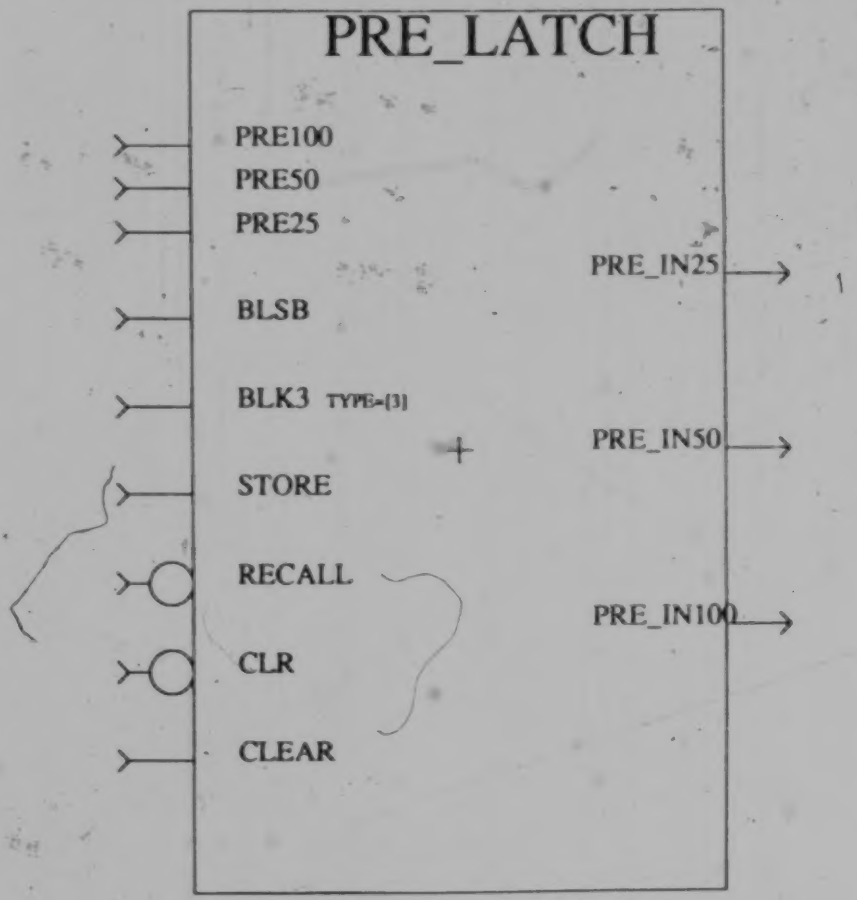
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: LATCH5 DATE: MAY 9 1989 PAGE: 21 OF 63



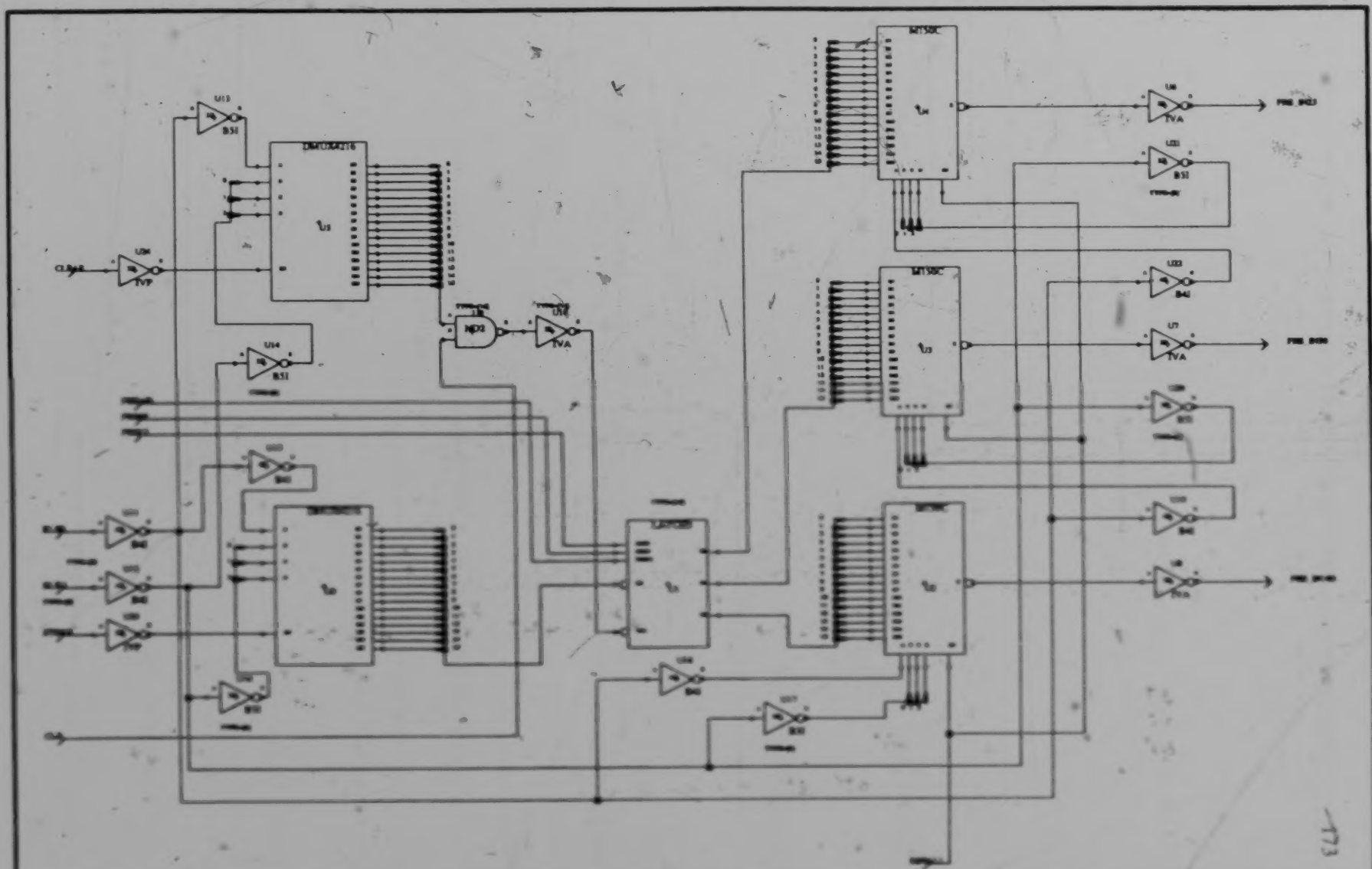
U OF A (E.E.)	DRAWN BY: M. PARANJAPE	TITLE: M.SC. PROJECT	MODULE: UDC8
	CHK'D:	A DATA ACQUISITION GATE ARRAY DESIGN	DATE: MAY 9 1989
			PAGE: 22 OF 63



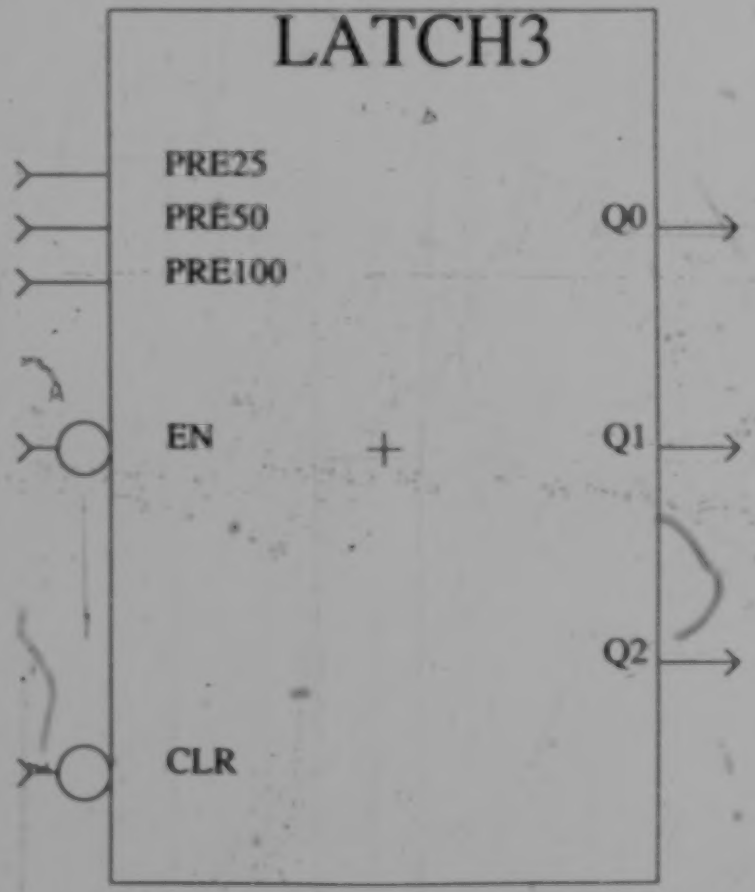
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: UDC8 DATE: MAY 9 1989 PAGE: 23 OF 63



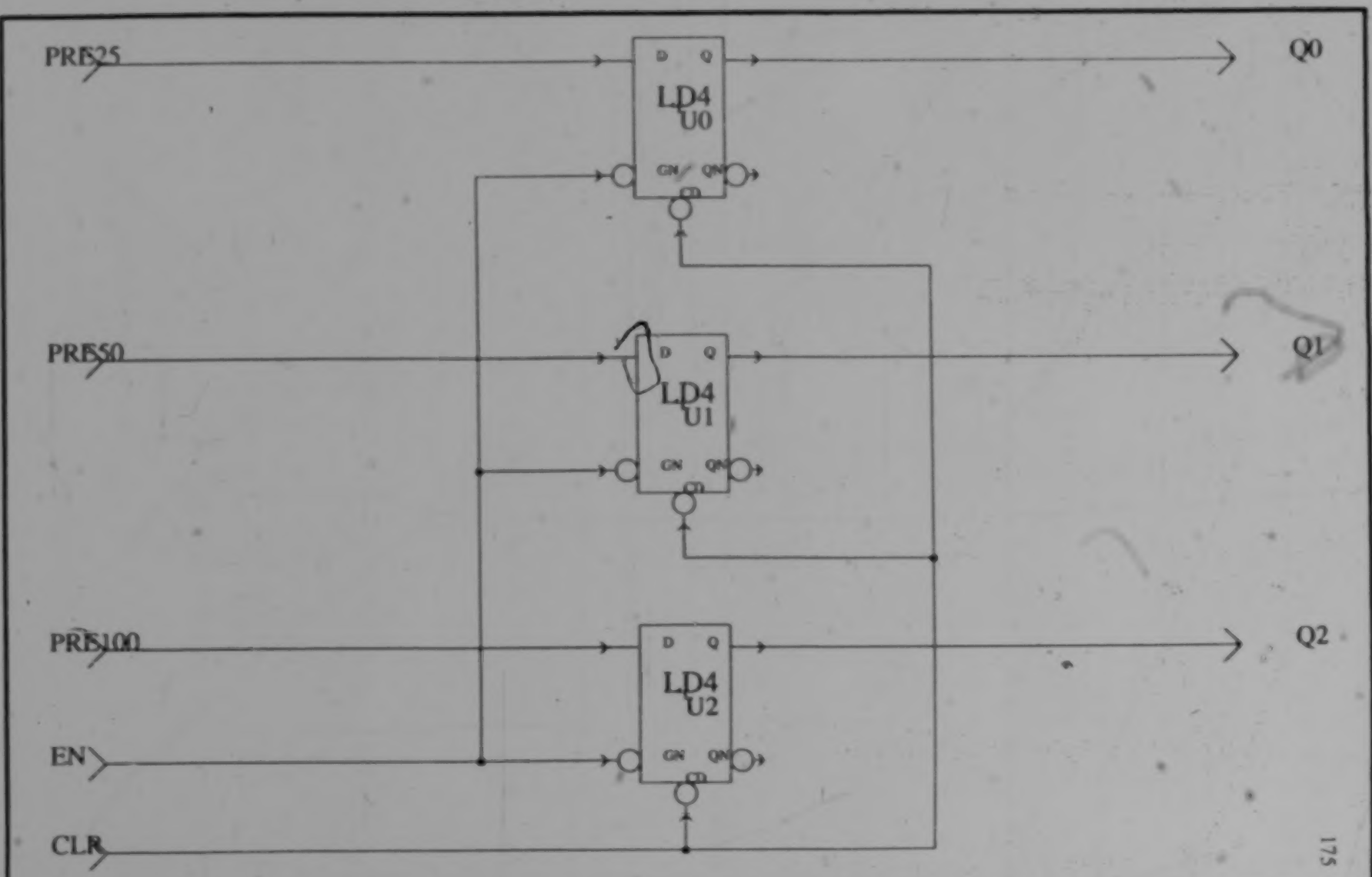
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: PRE_LATC DATE: MAY 9 1989 PAGE: 24 OF 63



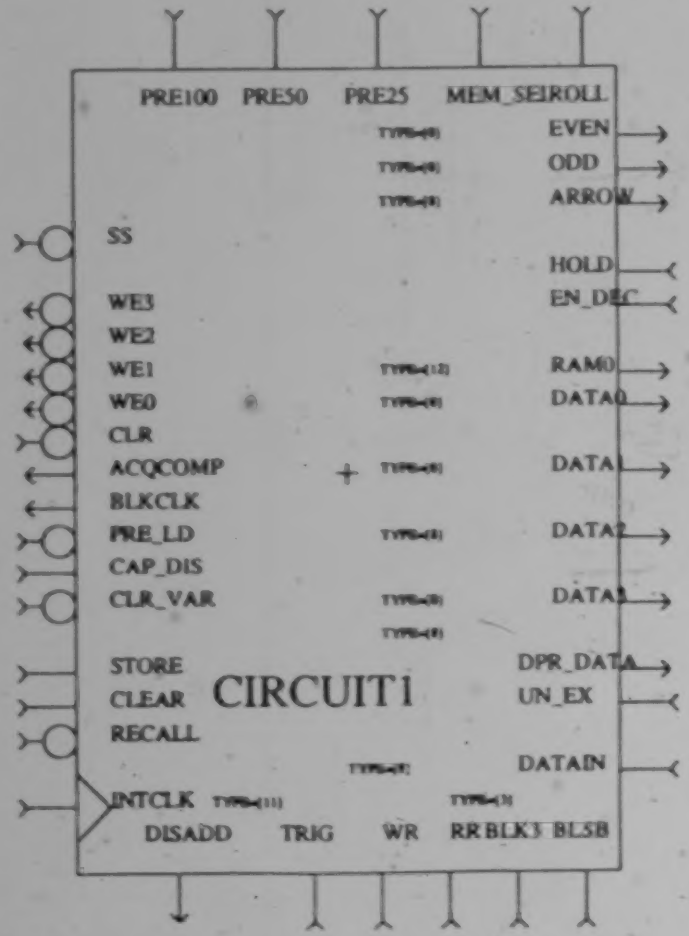
U OF A (E.E.)	DRAWN BY: M. PARANJAPE	TITLE: M.SC. PROJECT	MODULE: PRE_LATC
	CHK'D:	A DATA ACQUISITION GATE ARRAY DESIGN	DATE: MAY 9 1989
			PAGE: 25 OF 63



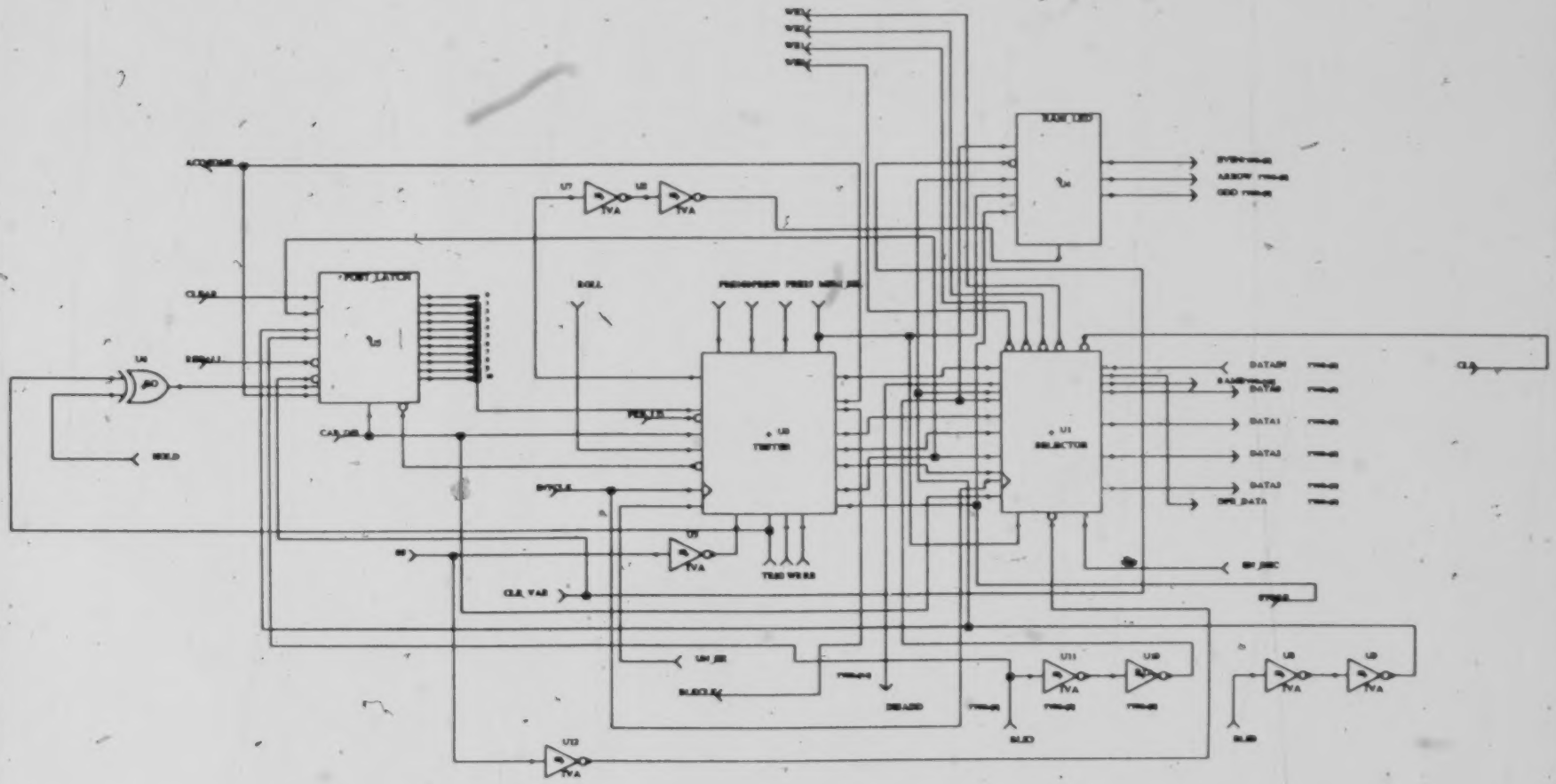
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE ARRAY DESIGN	MODULE: LATCH3 DATE: MAY 9 1989 PAGE: 26 OF 63



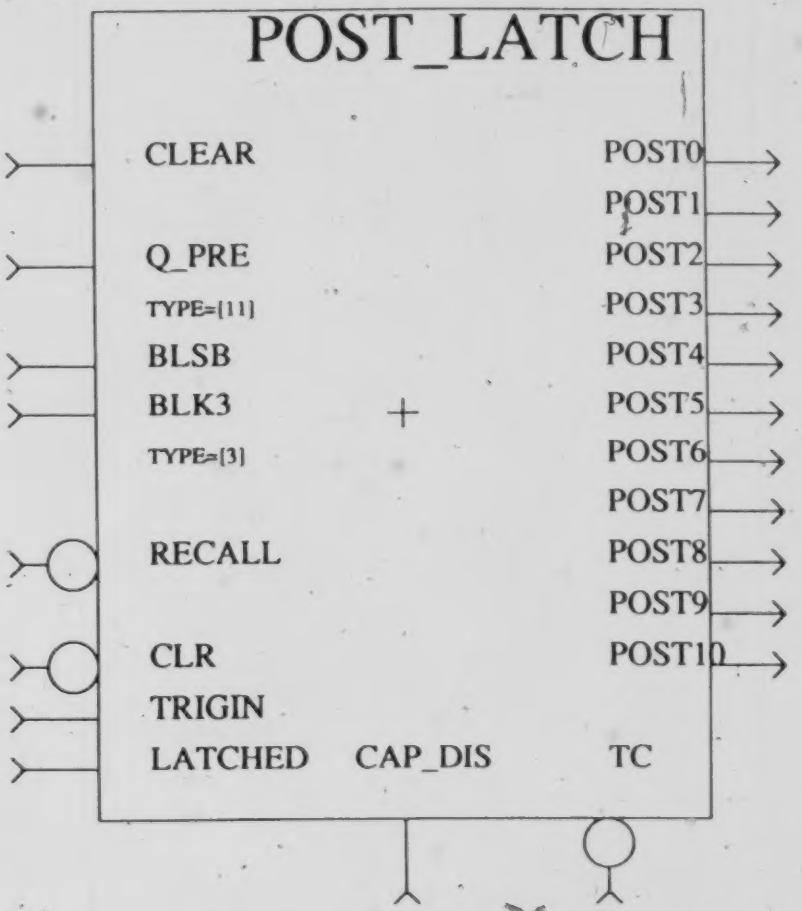
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE ARRAY DESIGN	MODULE: LATCH3 DATE: MAY 9 1989 PAGE: 27 OF 63



U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE ARRAY DESIGN	MODULE: CIRCUIT1 DATE: MAY 9 1989 PAGE: 28 OF 63

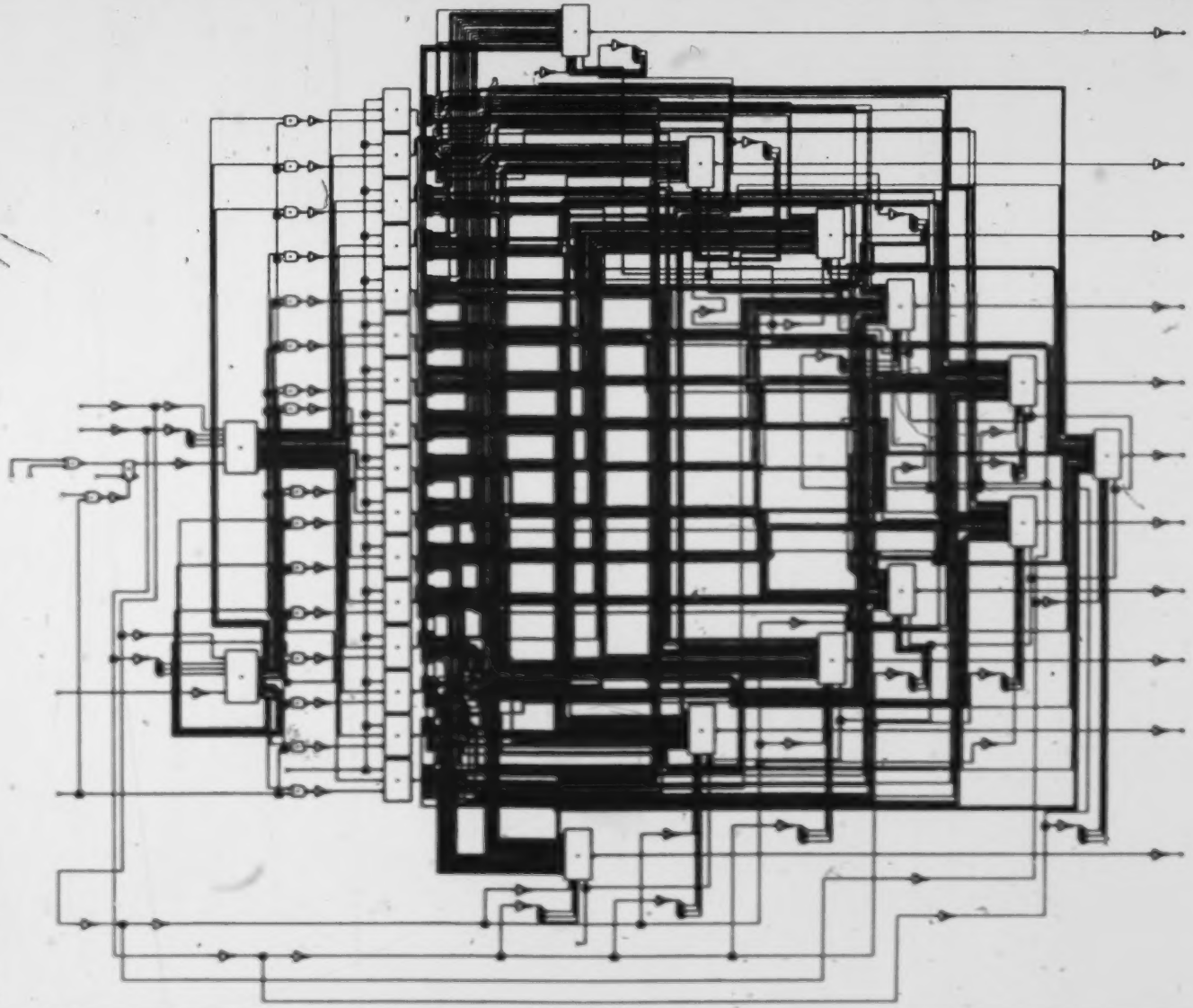


U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: CIRCUIT1 DATE: MAY 9 1989 PAGE: 29 OF 63

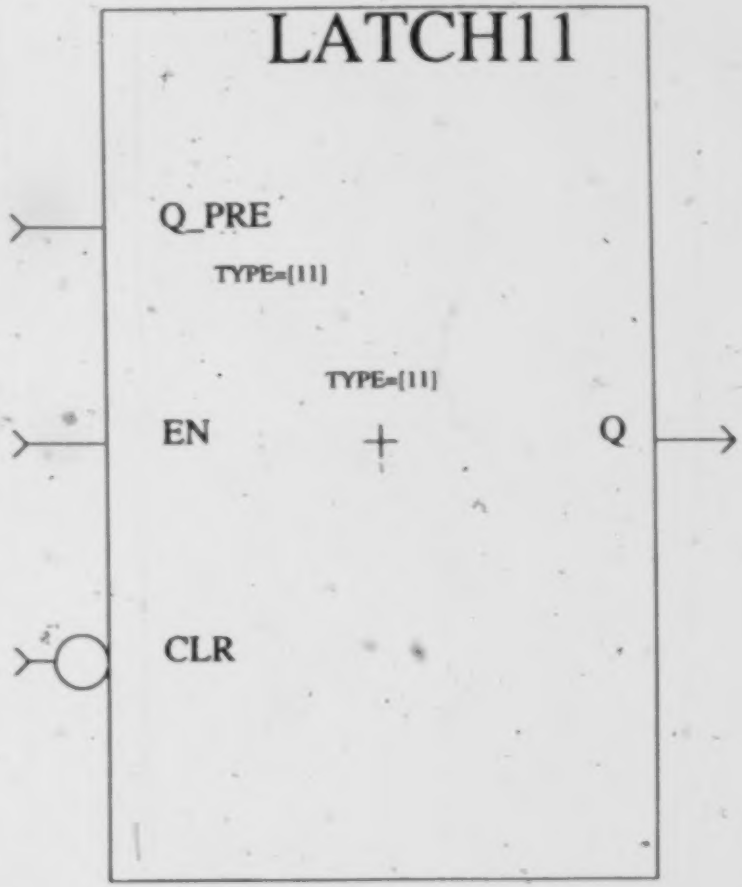


178

U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: POST_LAT DATE: MAY 9 1989 PAGE: 30 OF 63

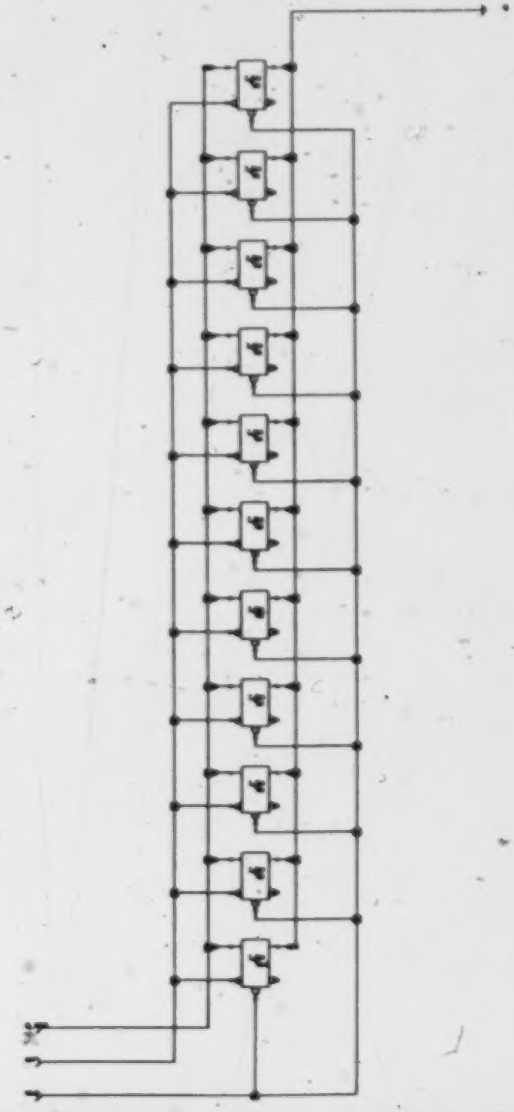


U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: POST_LAT DATE: MAY 9 1989 PAGE: 31 OF 63



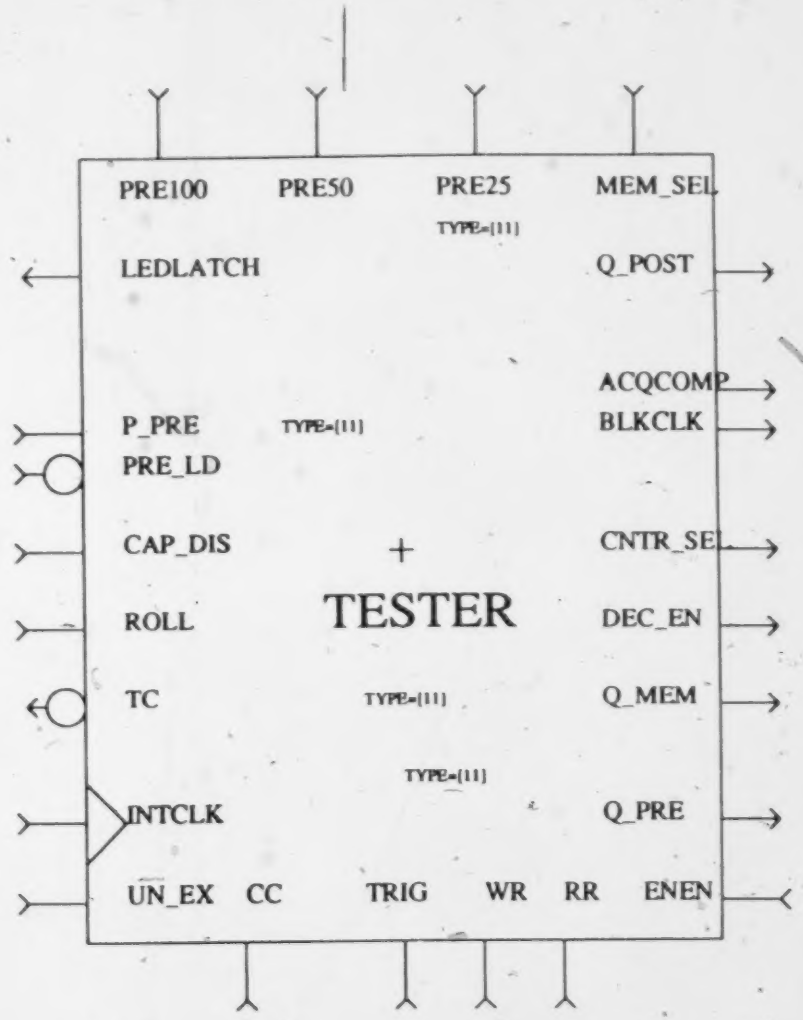
180

U OF A (E.E.)	DRAWN BY: M. PARANJAPE	TITLE: M.SC. PROJECT	MODULE: LATCH11
	CHK'D:	A DATA ACQUISITION GATE-ARRAY DESIGN	DATE: MAY 9 1989
			PAGE: 32 OF 63

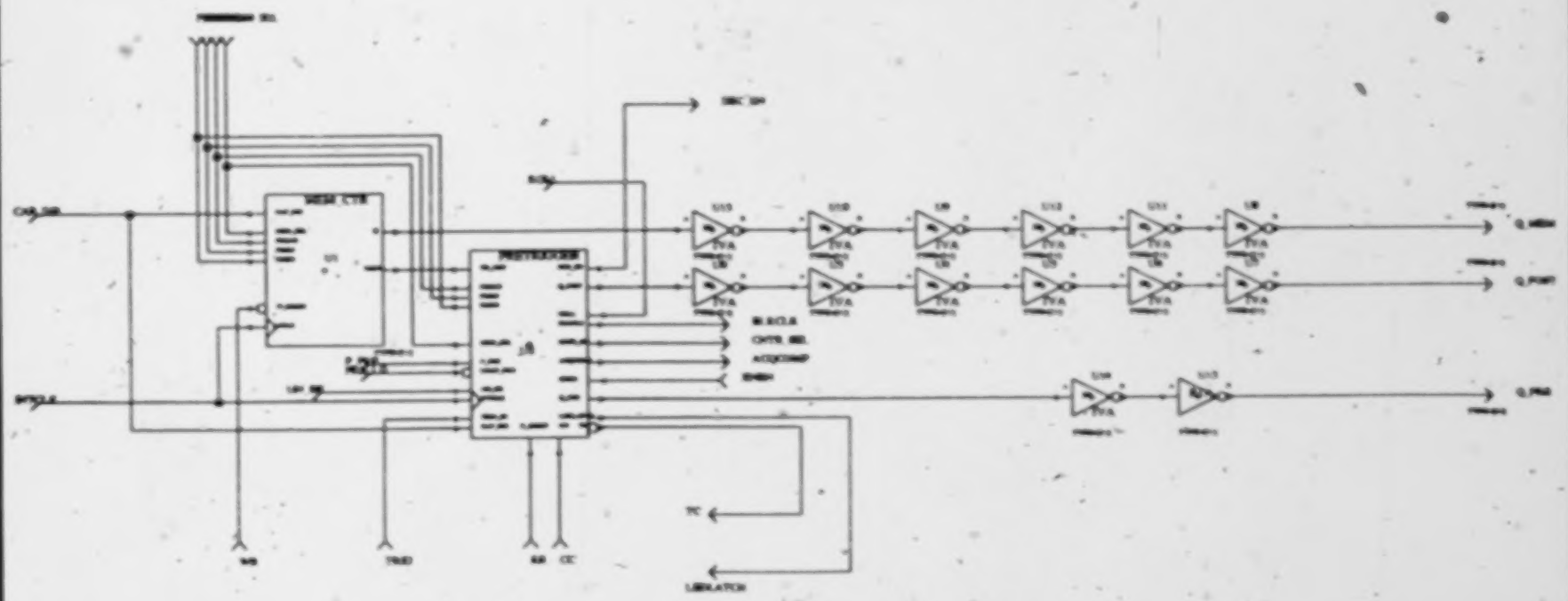


181

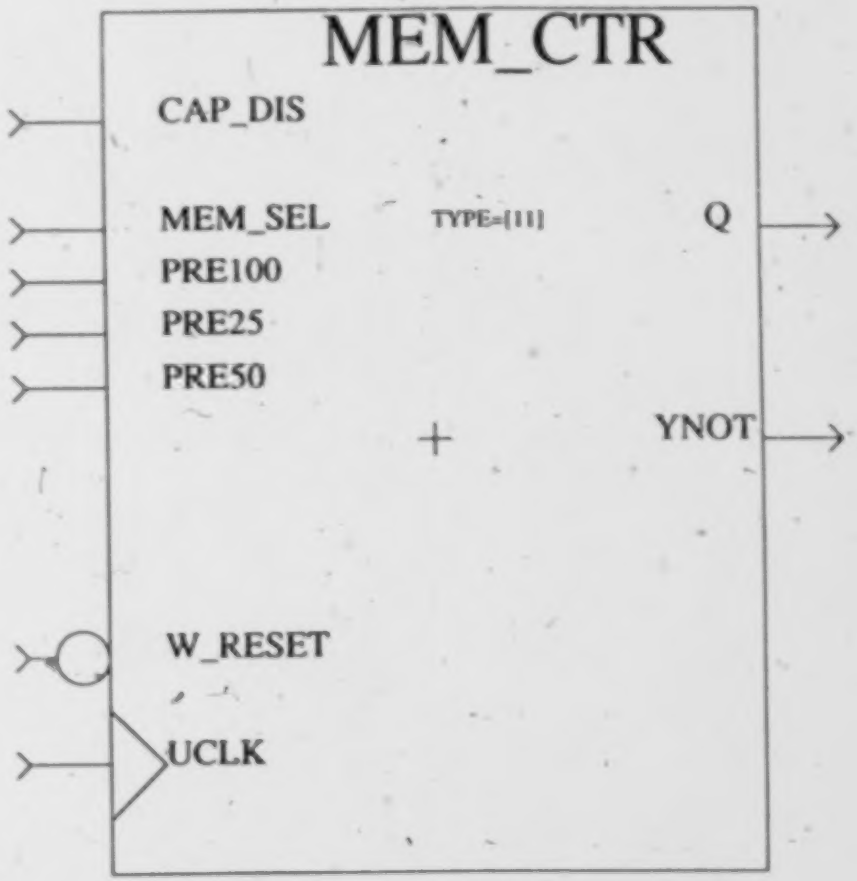
U OF A (E.E.)	DRAWN BY: M. PARANJAPE	TITLE: M.SC. PROJECT	MODULE: LATCH11
	CHK'D:	A DATA ACQUISITION GATE ARRAY DESIGN	DATE: MAY 9 1989
			PAGE: 33 OF 63



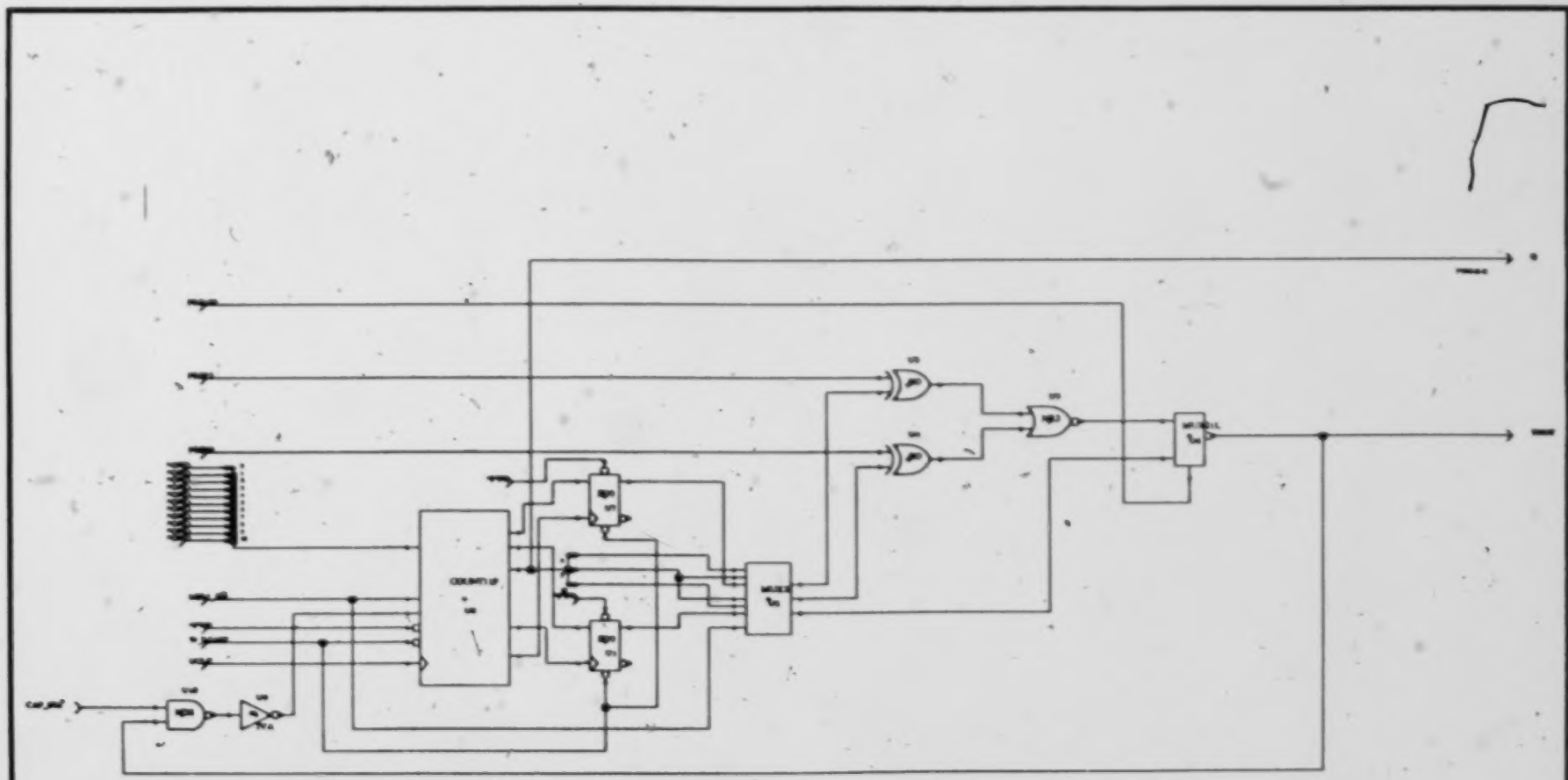
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE ARRAY DESIGN	MODULE: TESTER DATE: MAY 9 1989 PAGE: 34 OF 63



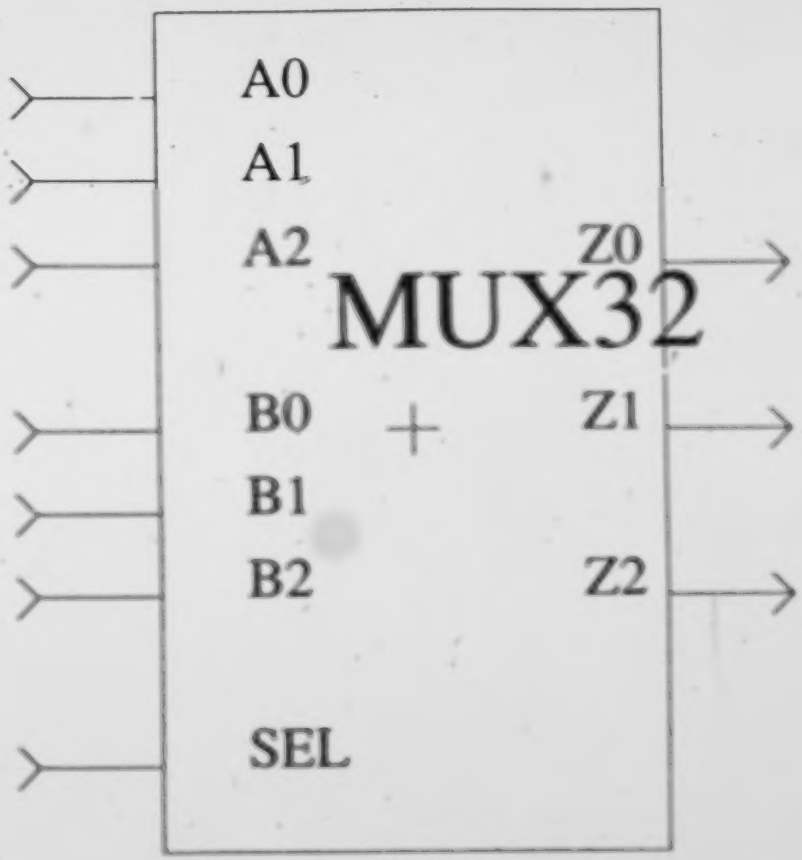
U OF A (E.E.)	DRAWN BY: M. PARANJAPE	TITLE: M.SC. PROJECT	MODULE: TESTER
	CHK'D:	A DATA ACQUISITION GATE ARRAY DESIGN	DATE: MAY 9 1989
			PAGE: 35 OF 63



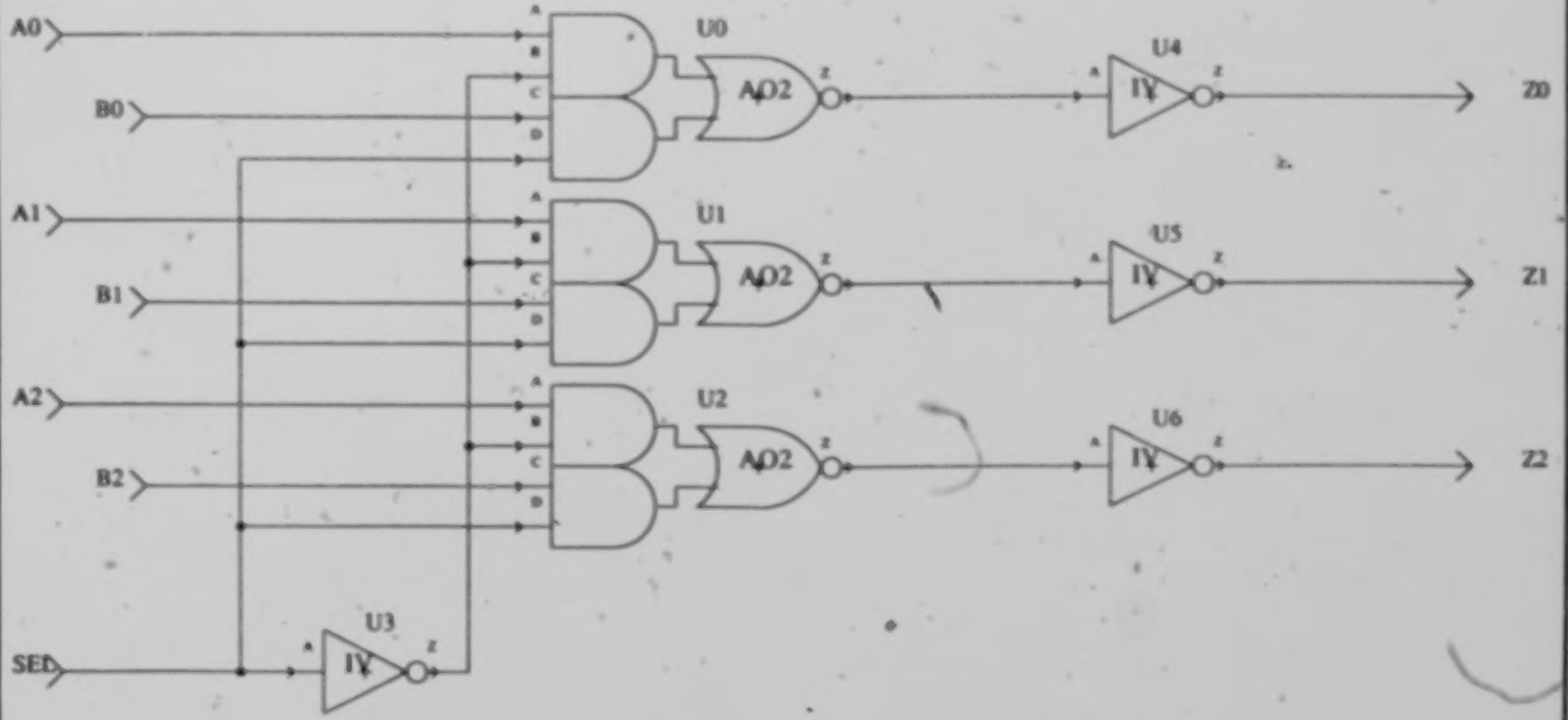
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: MEM_CTR DATE: MAY 9 1989 PAGE: 36 OF 63



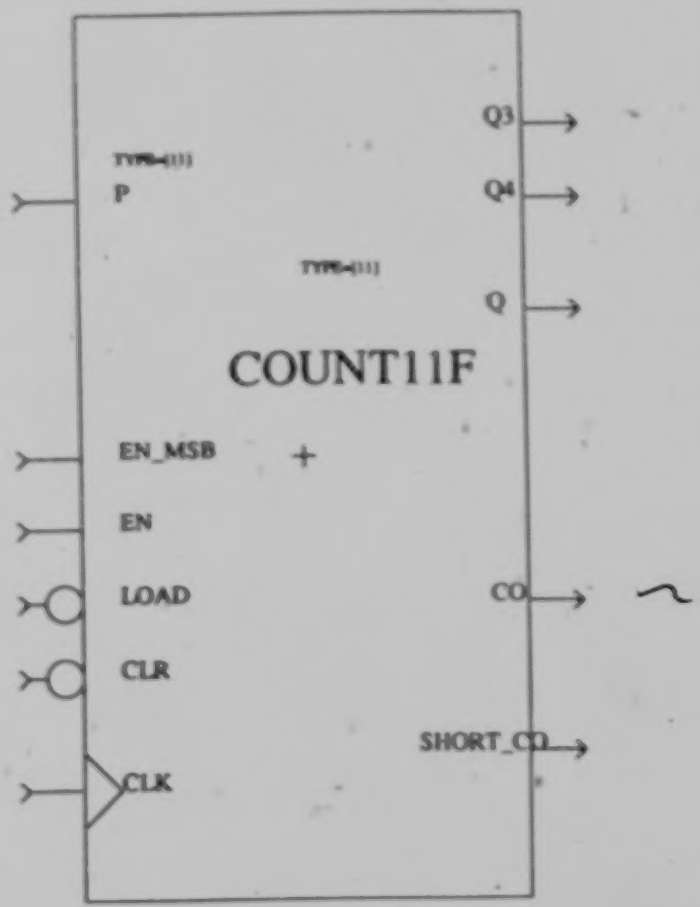
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: MEM_CTR DATE: MAY 9 1989 PAGE: 37 OF 63



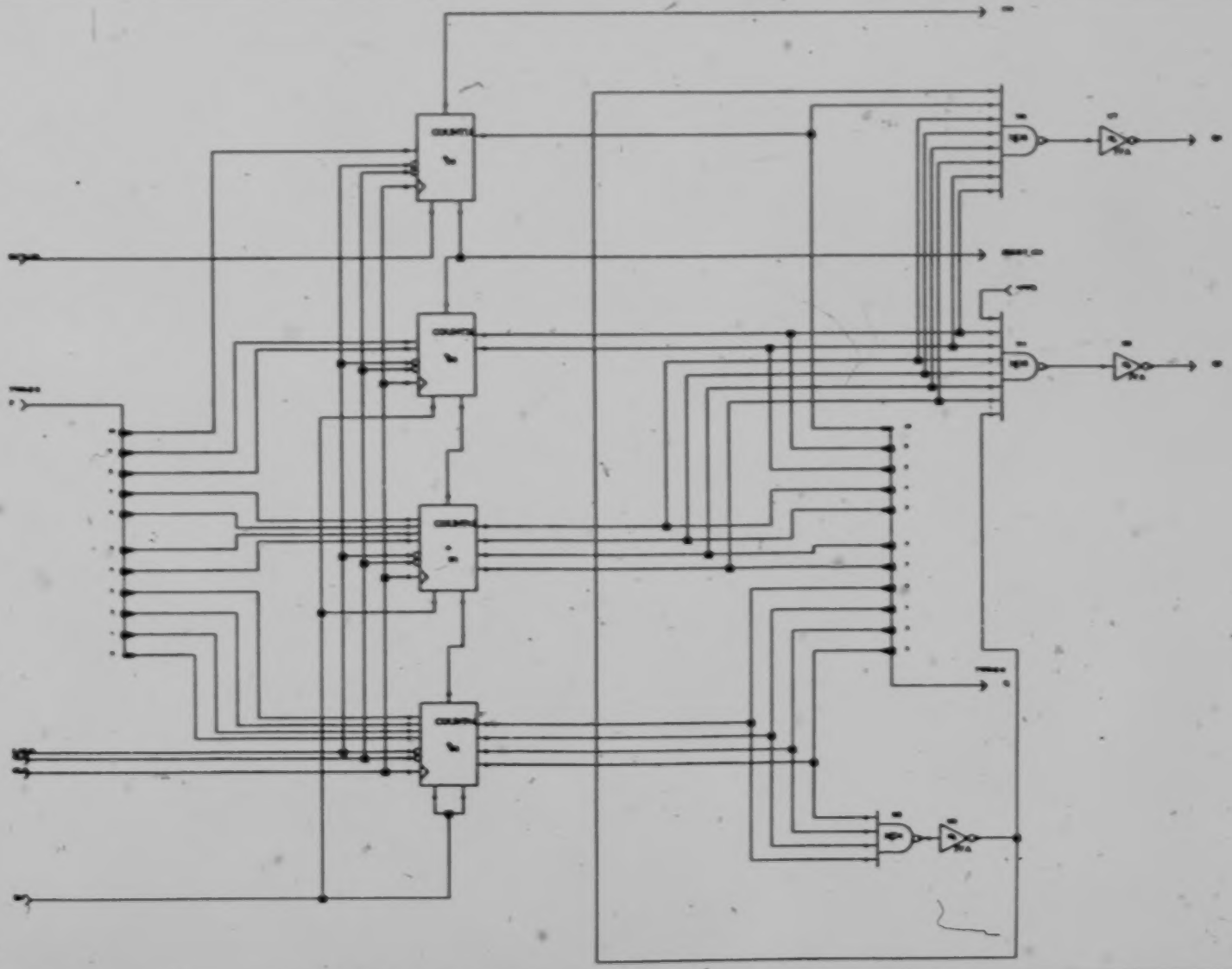
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE ARRAY DESIGN	MODULE: MUX32 DATE: MAY 9 1989 PAGE: 38 OF 63



U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE ARRAY DESIGN	MODULE: MUX32 DATE: MAY 9 1989 PAGE: 39 OF 63

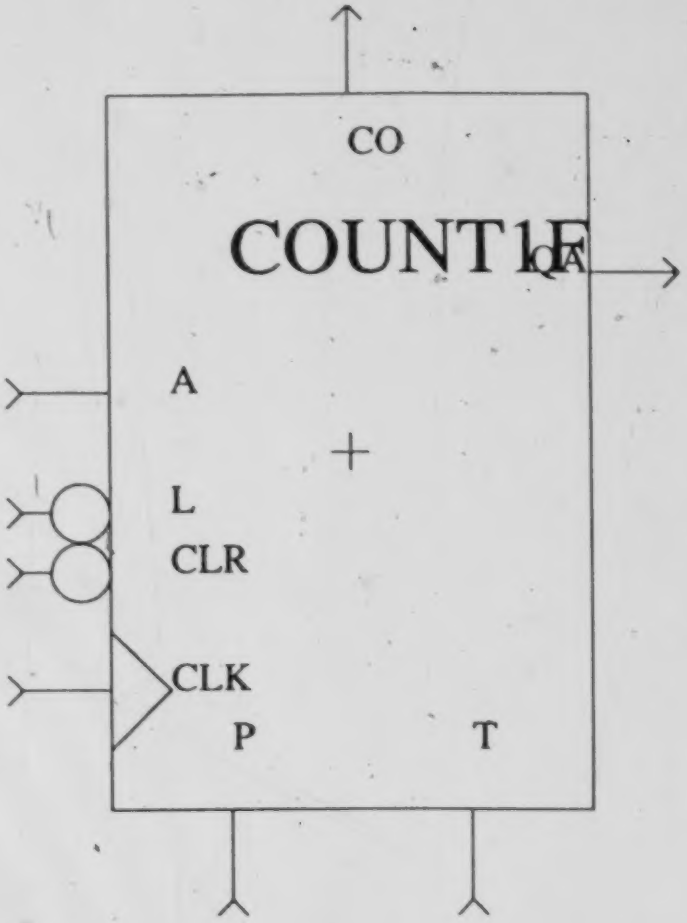


U OF A (E.E.)	DRAWN BY: M. PARANJAPE	TITLE: M.SC. PROJECT	MODULE: COUNT11F
	CHK'D:	A DATA ACQUISITION GATE-ARRAY DESIGN	DATE: MAY 9 1989
			PAGE: 40 OF 63



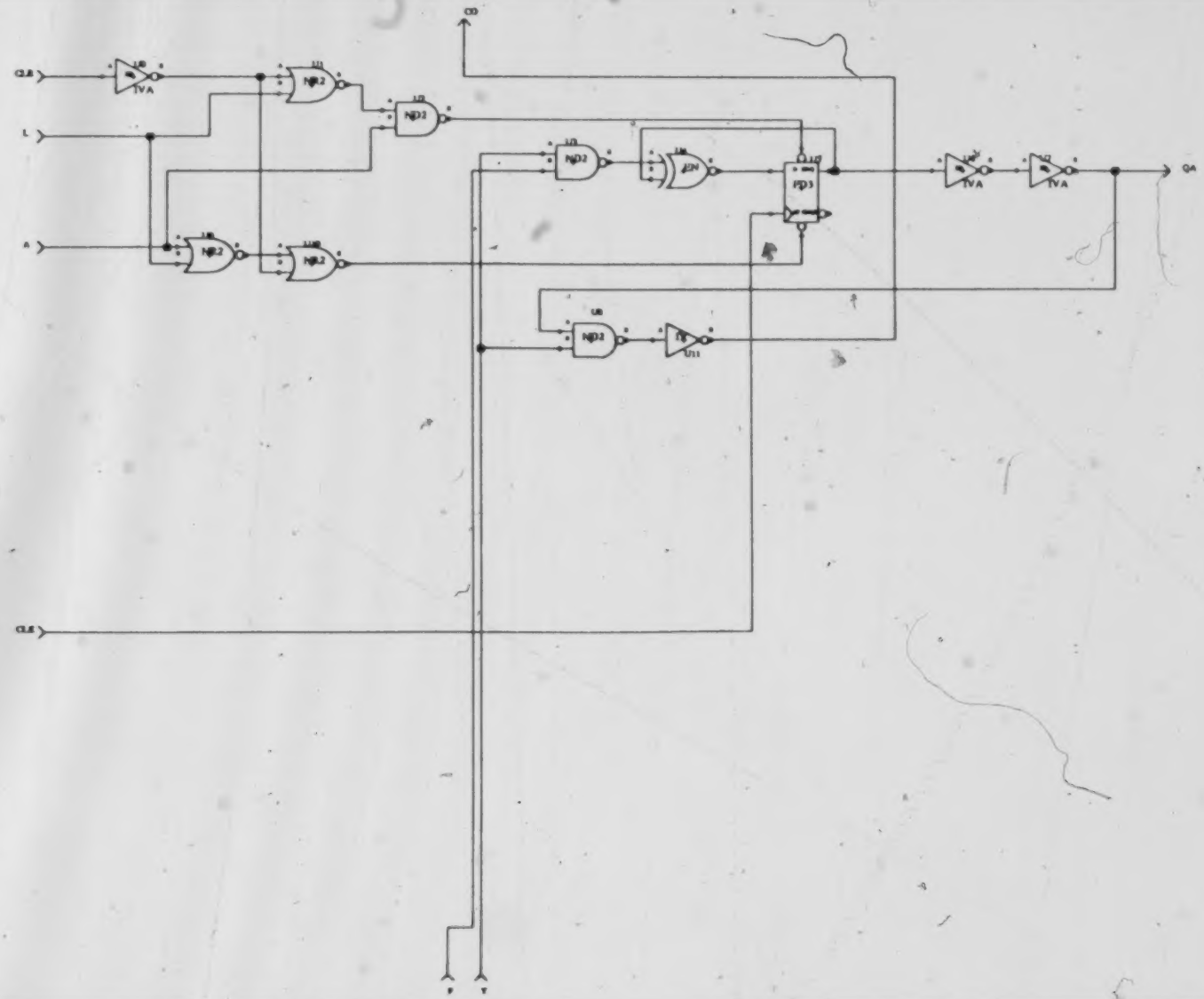
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: COUNT11F DATE: MAY 9 1989 PAGE: 41 OF 63

681

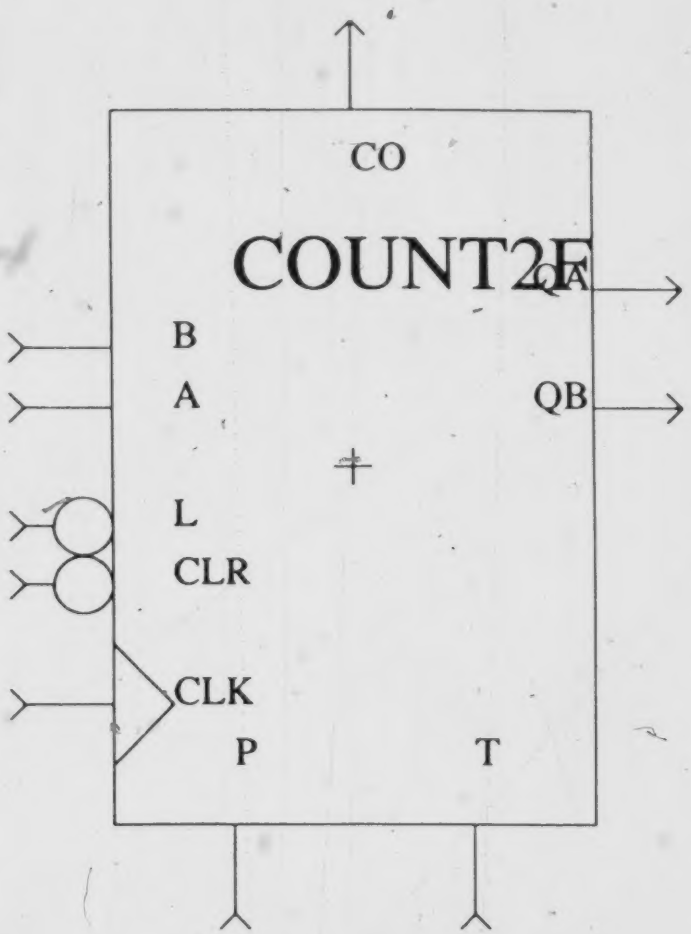


190

U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: COUNT1F DATE: MAY 9 1989 PAGE: 42 OF 63

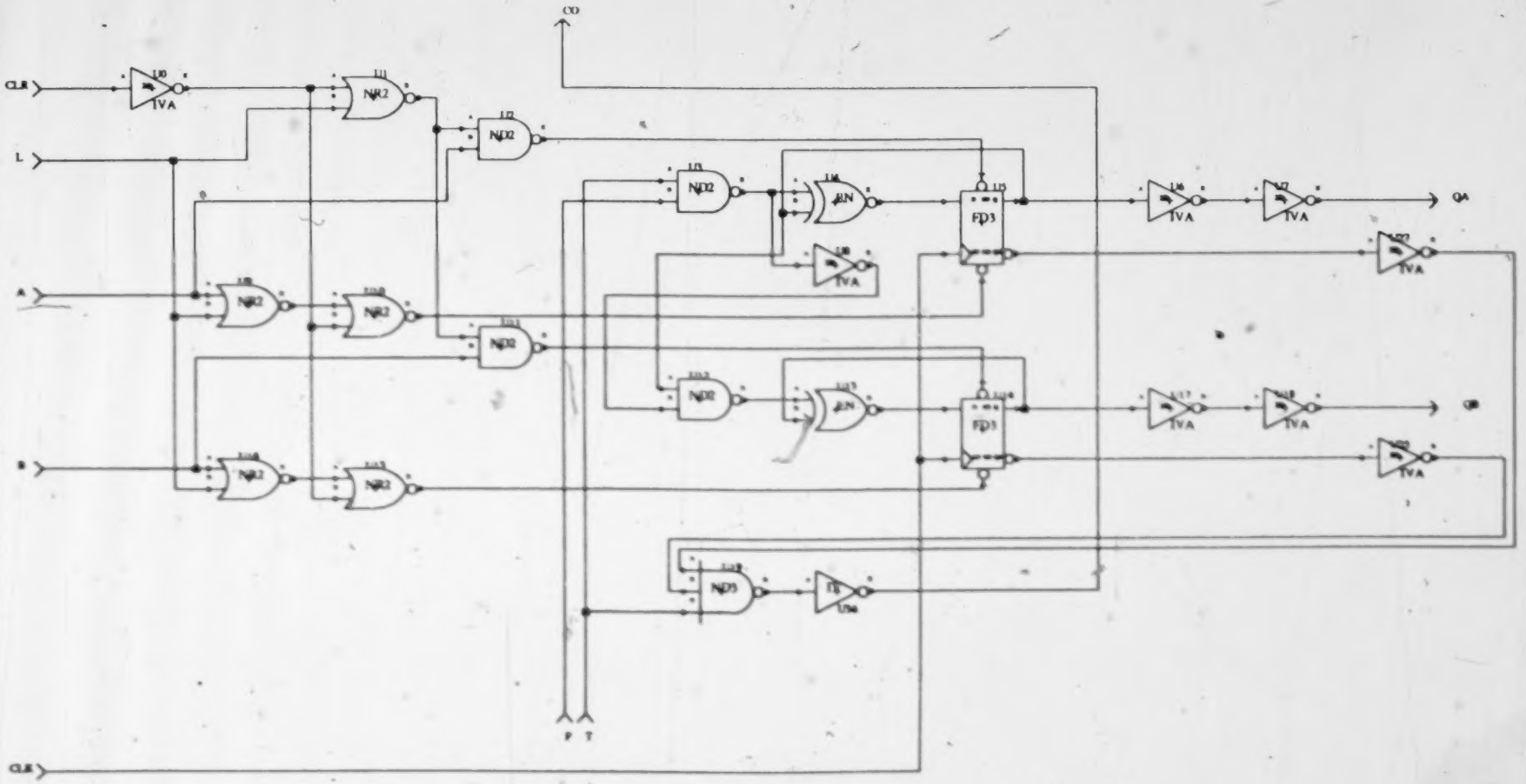


U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: COUNT1F DATE: MAY 9 1989 PAGE: 43 OF 63

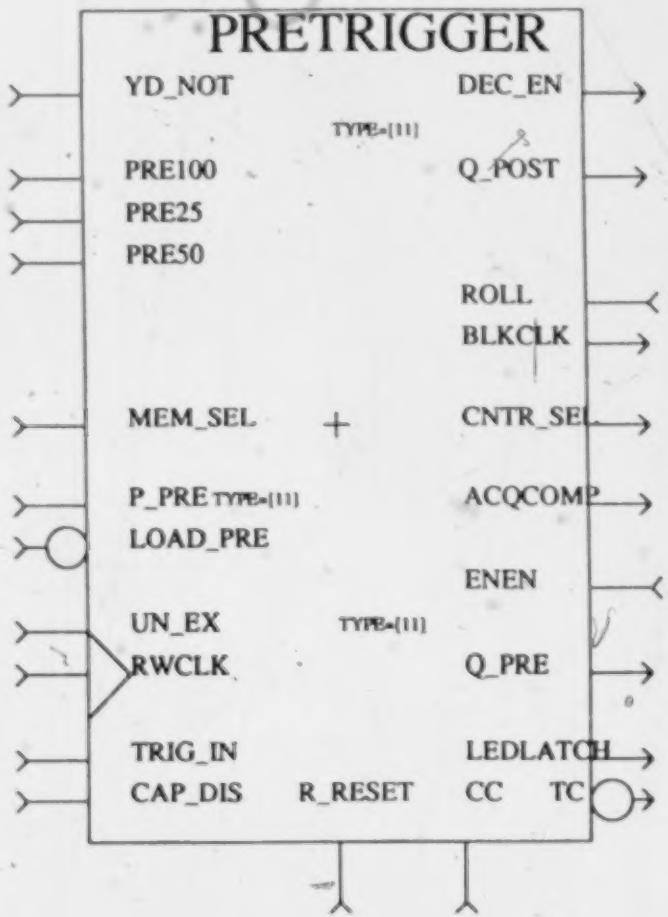


192

U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: COUNT2F DATE: MAY 9 1989 PAGE: 44 OF 63

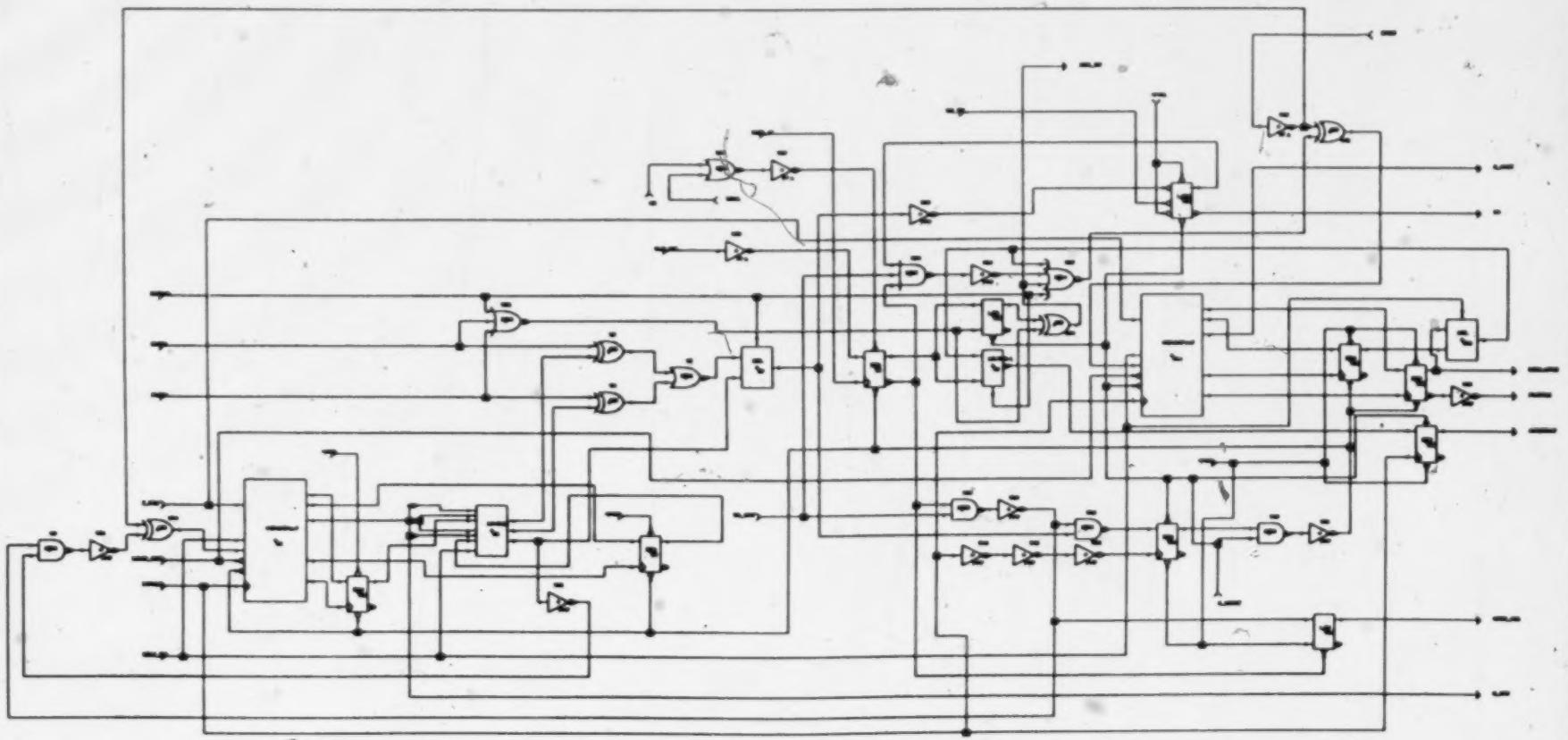


U OF A (E.E.)	DRAWN BY: M. PARANJAPE	TITLE: M.SC. PROJECT	MODULE: COUNT2P
	CHK'D:	A DATA ACQUISITION GATE ARRAY DESIGN	DATE: MAY 9 1989
			PAGE: 45 OF 63

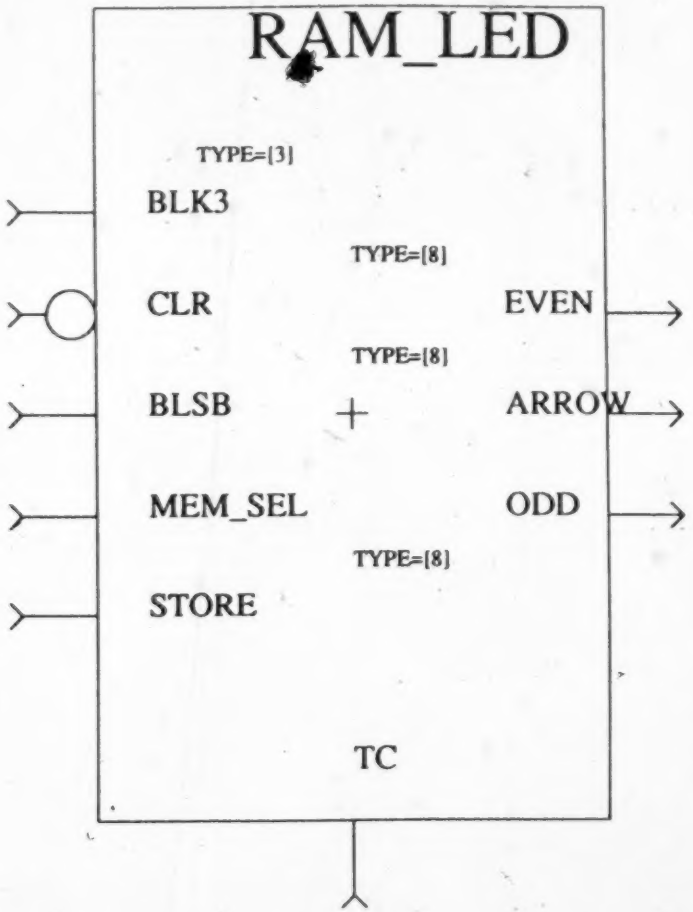


194

U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: PRETRIGG DATE: MAY 9 1989 PAGE: 46 OF 63

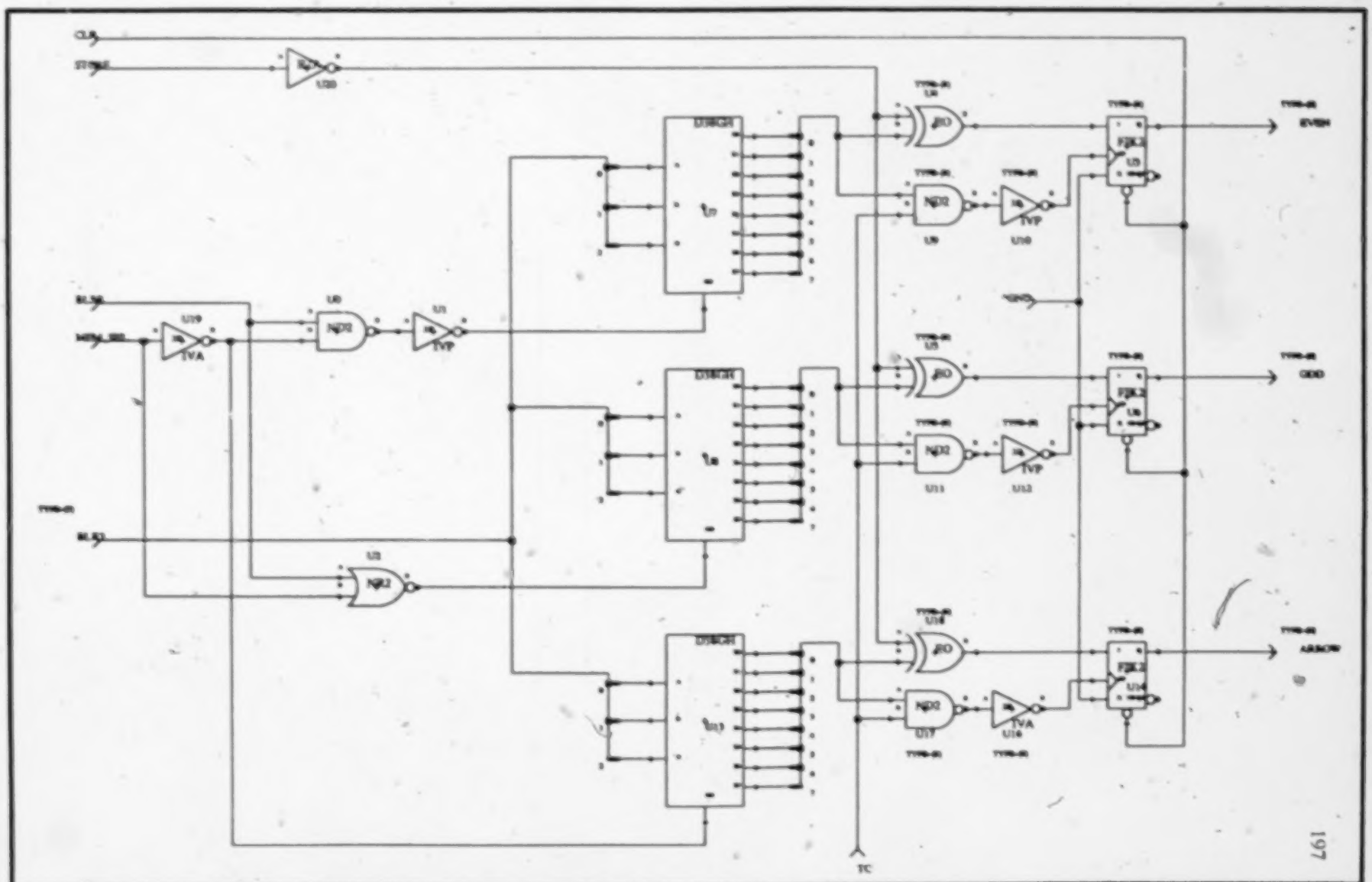


U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: PRETRIGG DATE: MAY 9 1989 PAGE: 47 OF 63

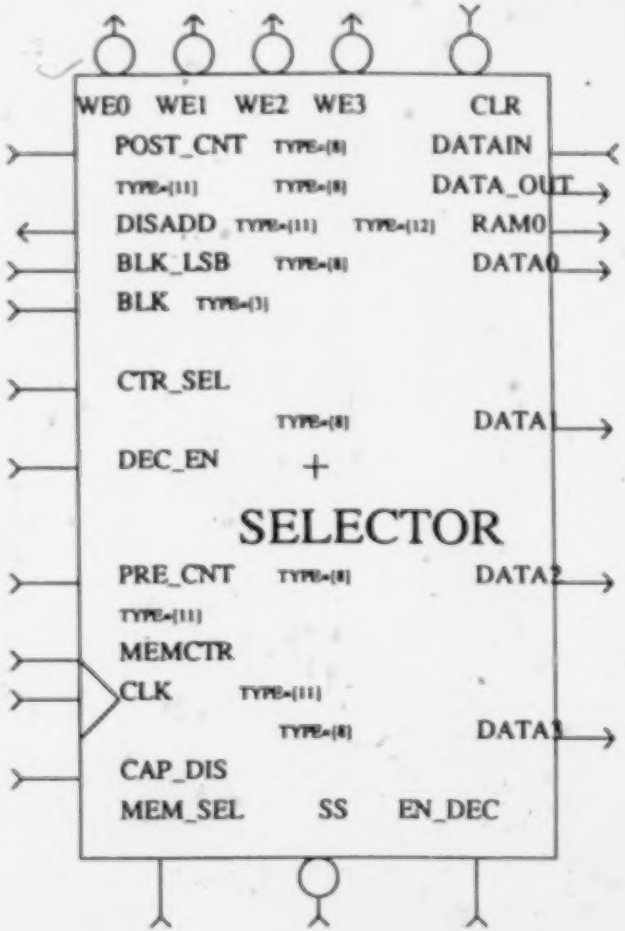


196

U OF A (E.E.)	DRAWN BY: M. PARANJAPE	TITLE: M.SC. PROJECT	MODULE: RAM_LED
	CHK'D:	A DATA ACQUISITION GATE ARRAY DESIGN	DATE: MAY 9 1989
			PAGE: 48 OF 63

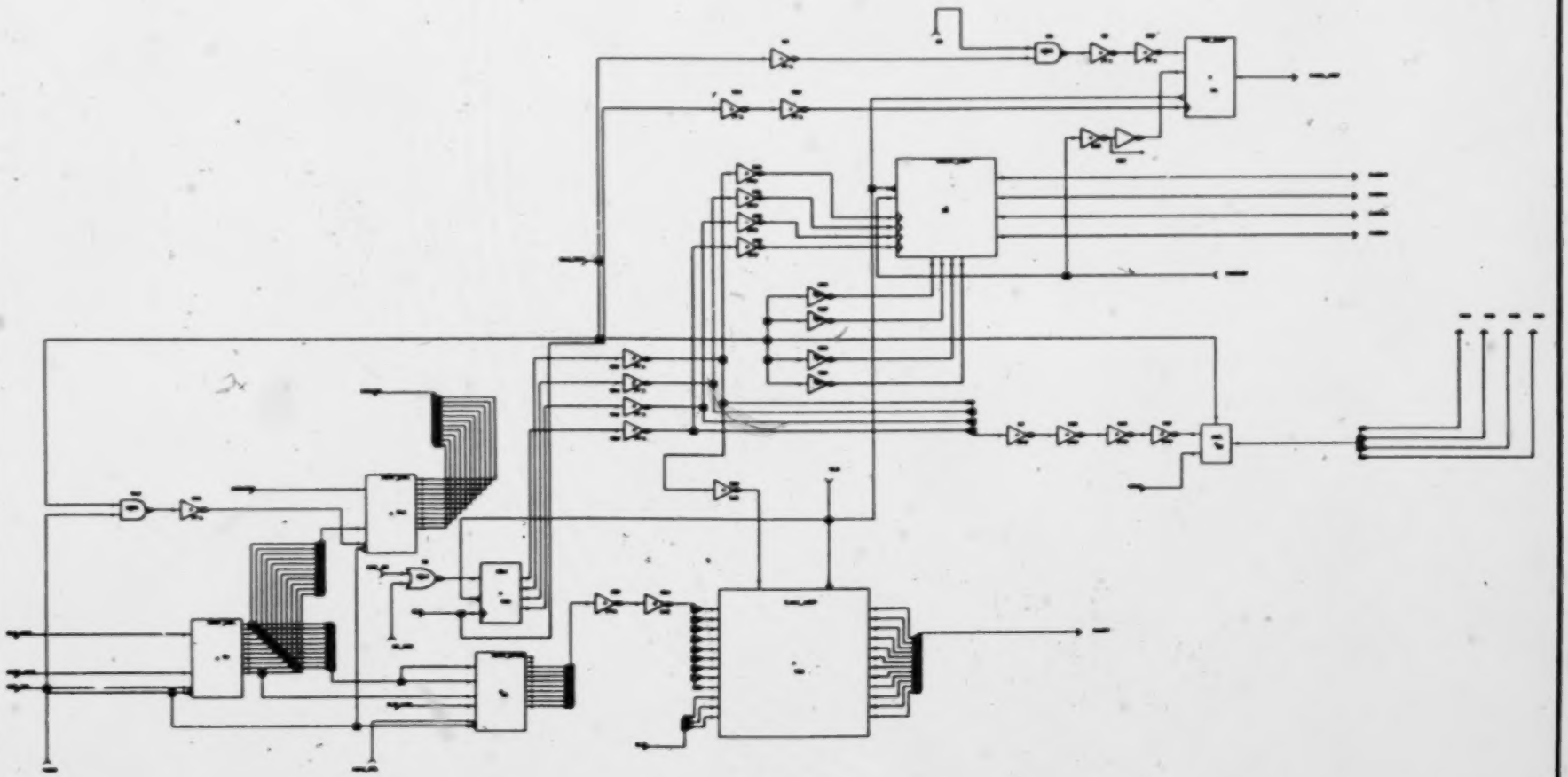


U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: RAM_LED DATE: MAY 9 1989 PAGE: 49 OF 63



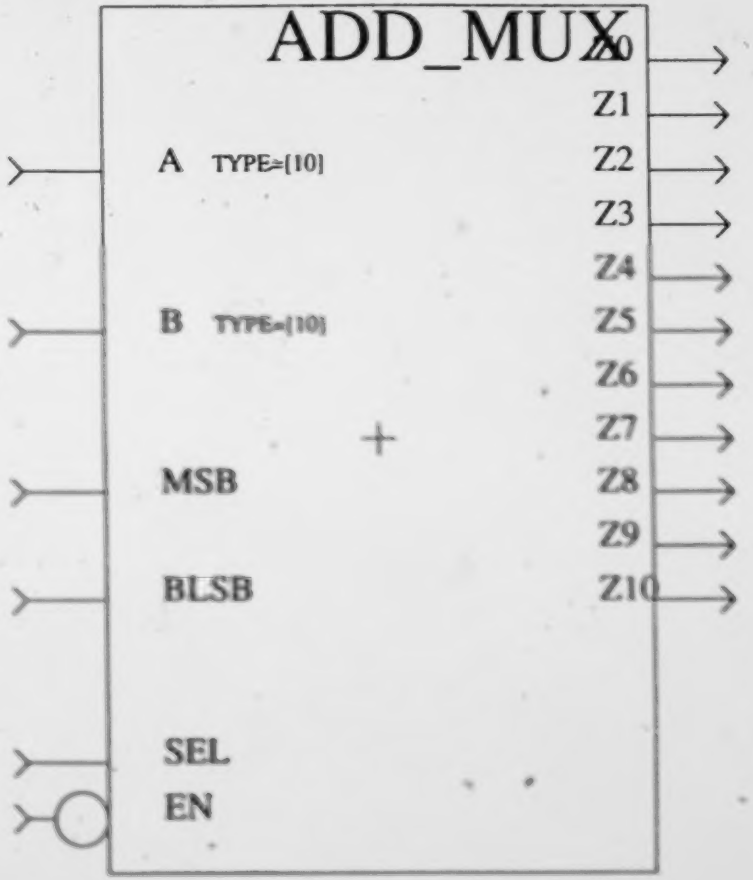
198

U OF A (E.E.)	DRAWN BY: M. PARANJAPE	TITLE: M.SC. PROJECT	MODULE: SELECTOR
	CHK'D:	A DATA ACQUISITION GATE-ARRAY DESIGN	DATE: MAY 9 1989
			PAGE: 50 OF 63

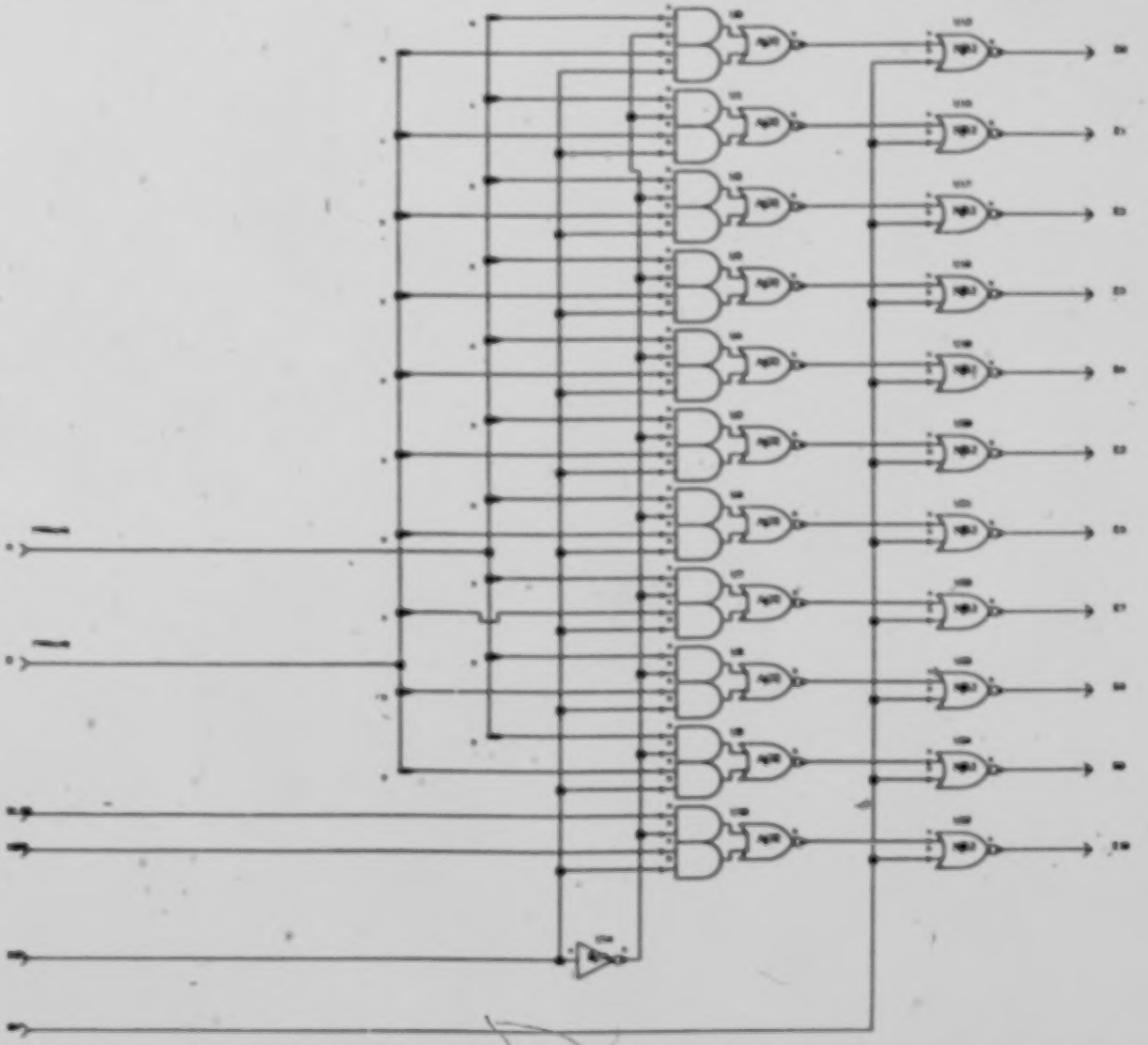


661

U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: SELECTOR DATE: MAY 9 1989 PAGE: 51 OF 63

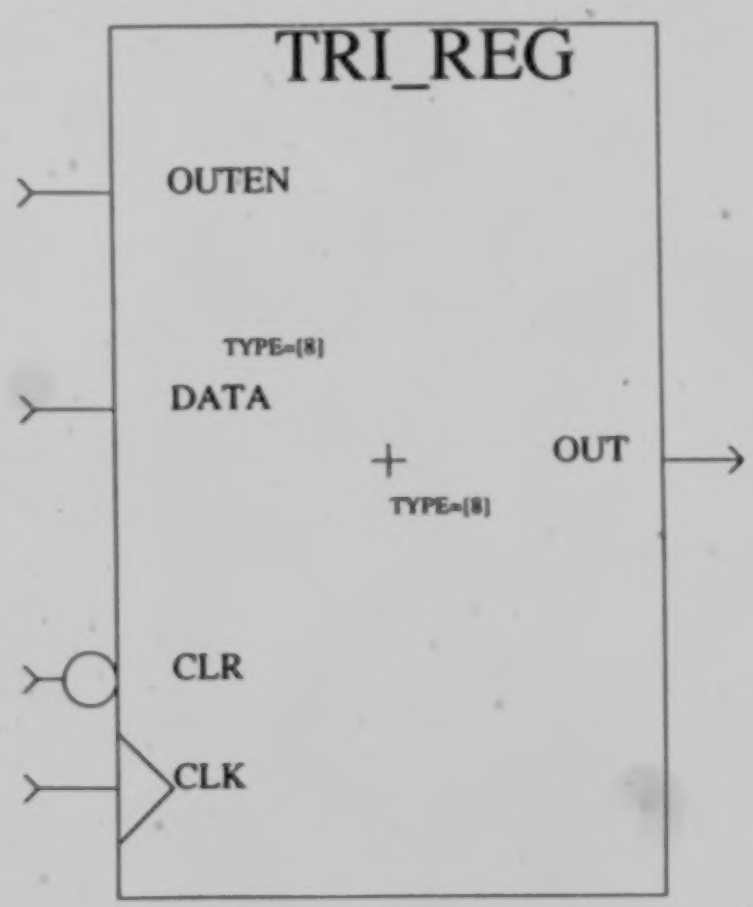


U OF A (E.E.)	DRAWN BY: M. PARANJAPE	TITLE: M.SC. PROJECT	MODULE: ADD_MUX
	CHK'D:	A DATA ACQUISITION GATE-ARRAY DESIGN	DATE: MAY 9 1989
			PAGE: 52 OF 63

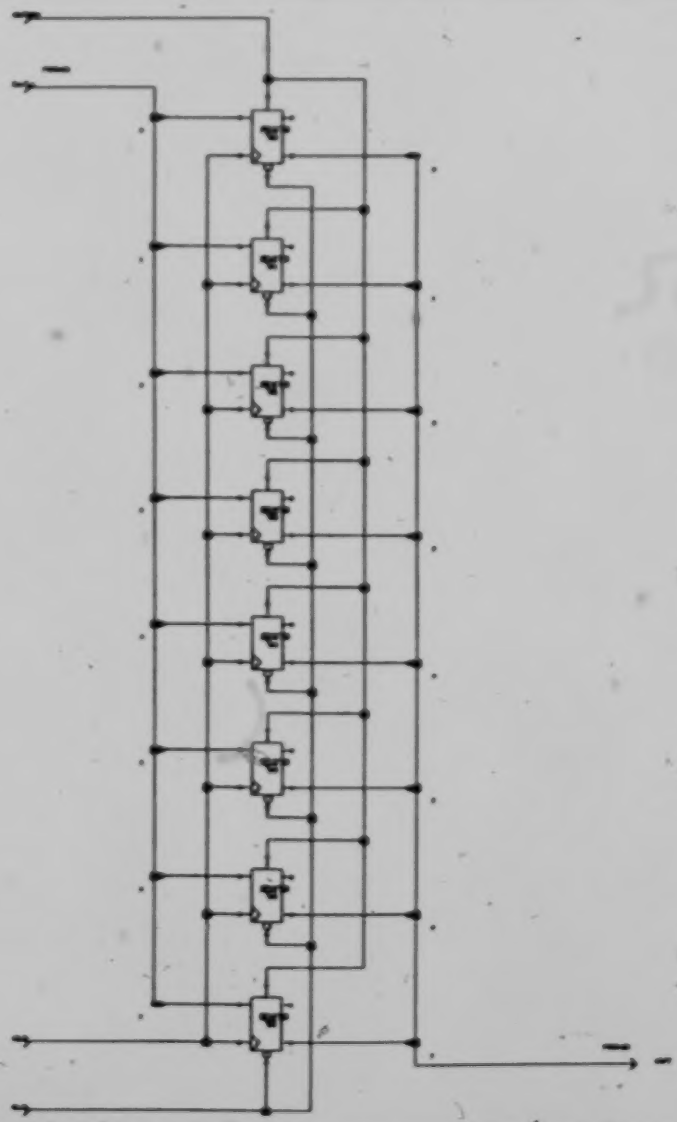


U OF A (E.E.)	DRAWN BY: M. PARANJAPE	TITLE: M.SC. PROJECT	MODULE: ADD_MUX
	CHK'D:	A DATA ACQUISITION GATE ARRAY DESIGN	DATE: MAY 9 1989
			PAGE: 53 OF 63

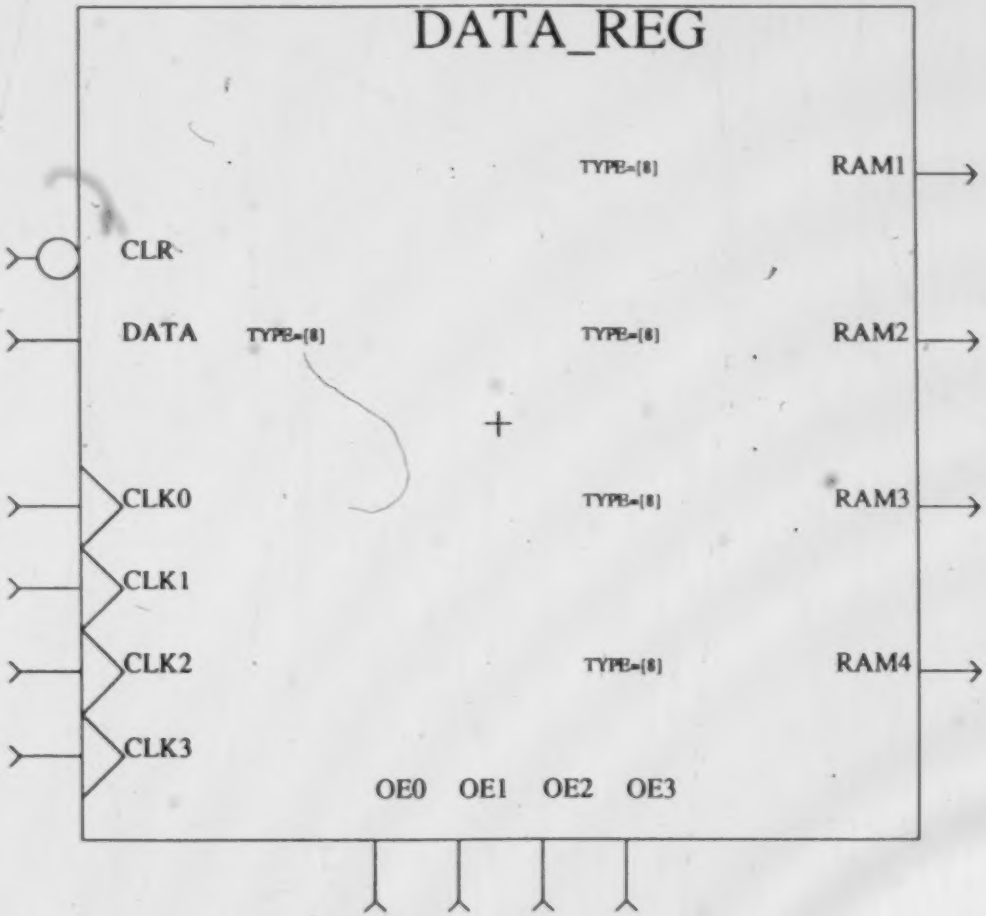
201



U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: TRI_REG DATE: MAY 9 1989 PAGE: 54 OF 63

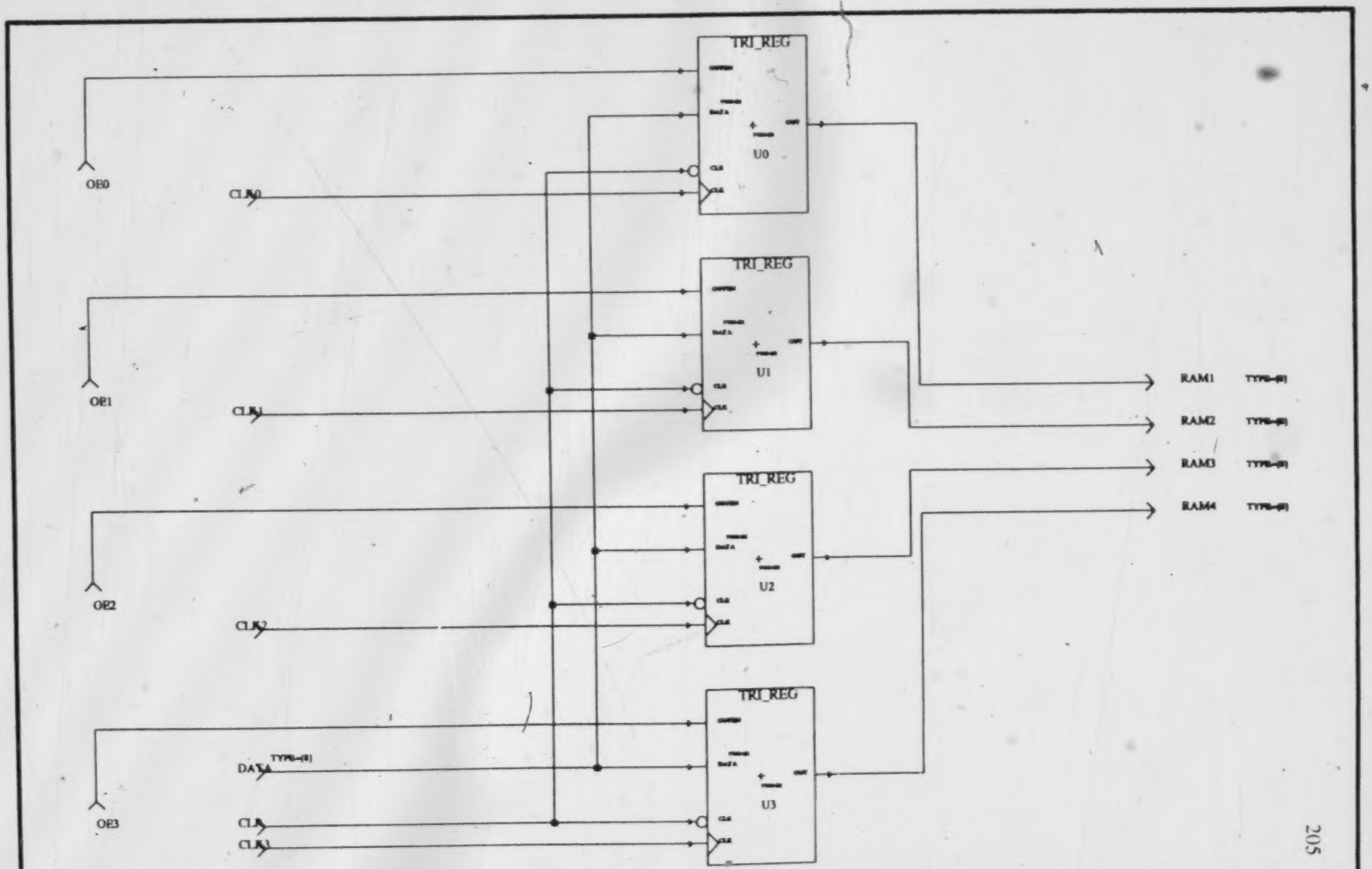


U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: TRI_REG DATE: MAY 9 1989 PAGE: 55 OF 63

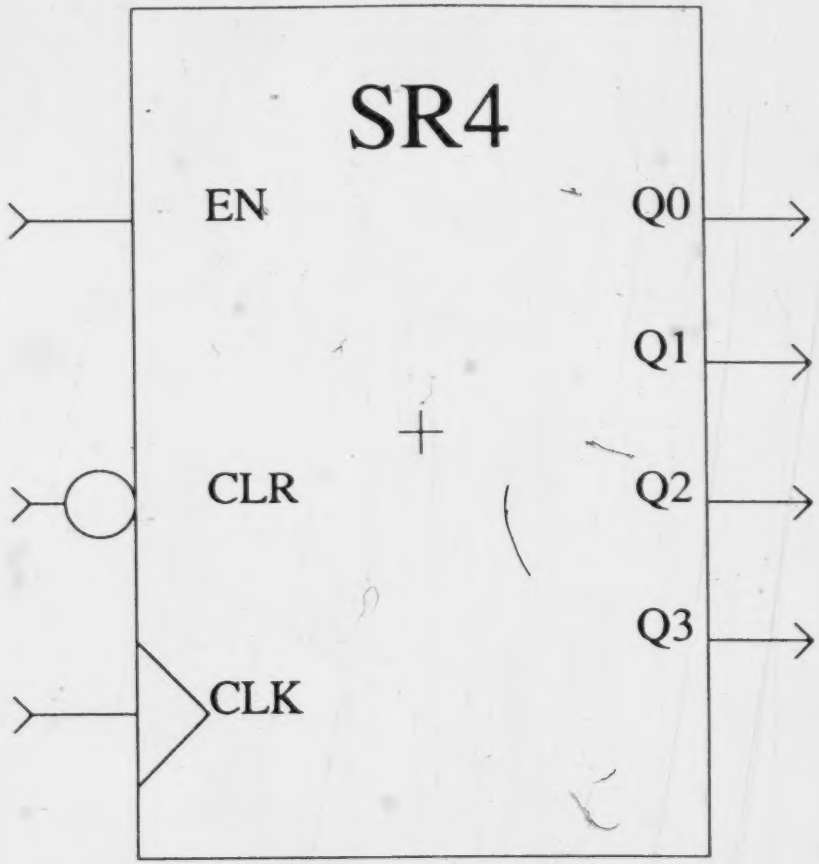


204

U OF A (E.E.)	DRAWN BY: M. PARANJAPE	TITLE: M.SC. PROJECT	MODULE: DATA_REG
	CHK'D:	A DATA ACQUISITION GATE-ARRAY DESIGN	DATE: MAY 9 1989
			PAGE: 56 OF 63

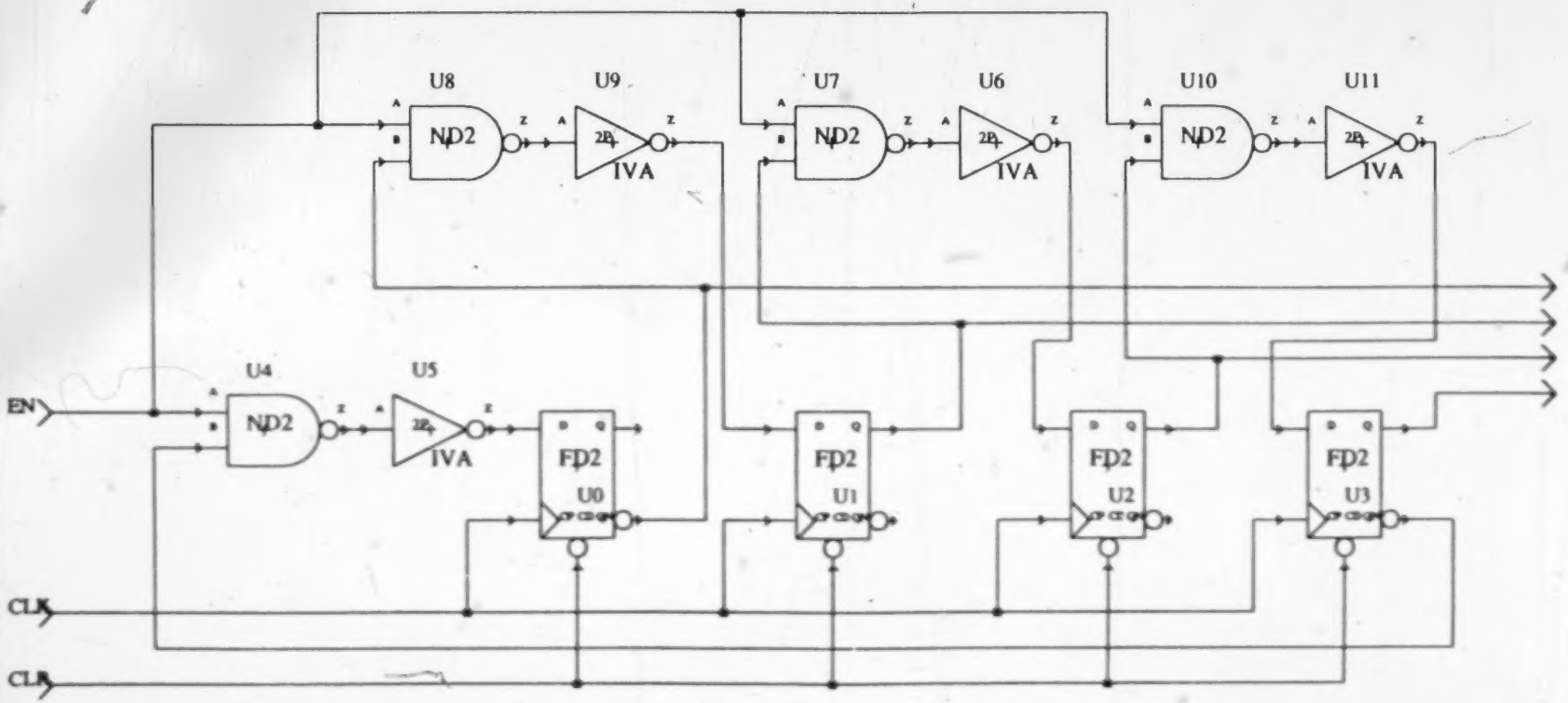


U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: DATA_REG DATE: MAY 9 1989 PAGE: 57 OF 63

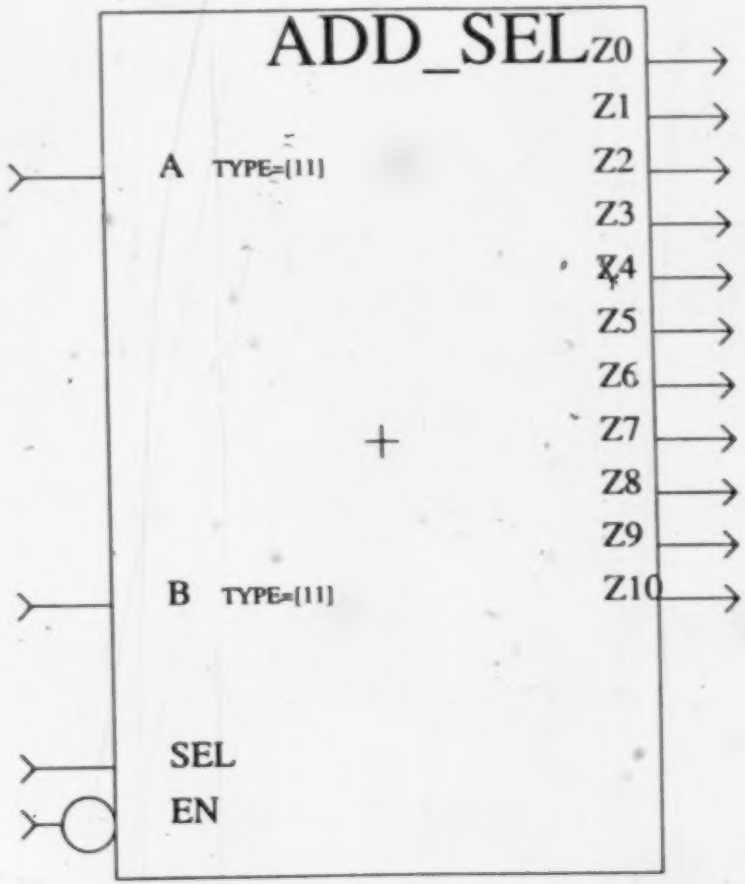


206

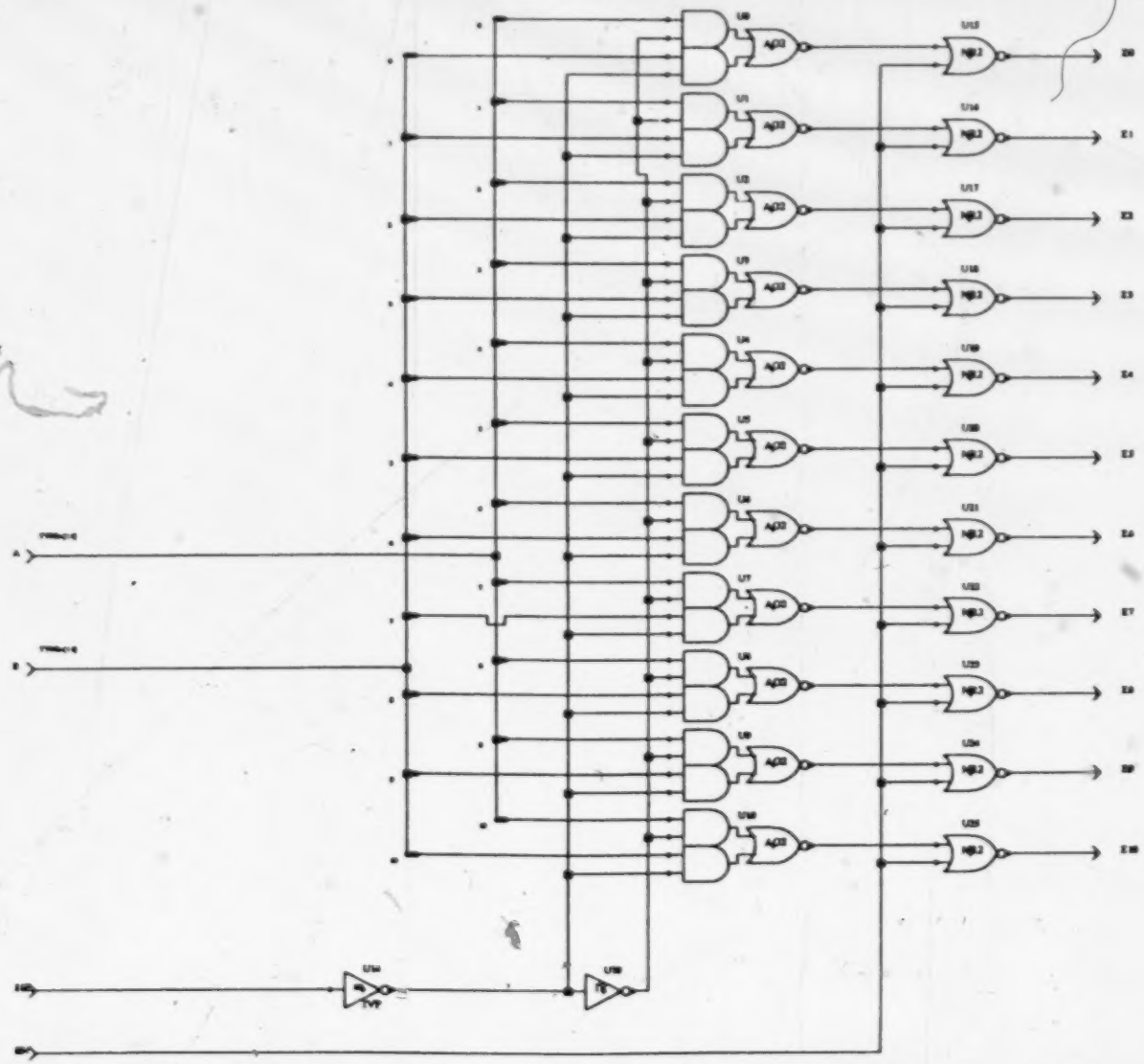
U OF A (E.E.)	DRAWN BY: M. PARANJAPE	TITLE: M.SC. PROJECT	MODULE: SR4
	CHK'D:	A DATA ACQUISITION GATE-ARRAY DESIGN	DATE: MAY 9 1989
			PAGE: 58 OF 63



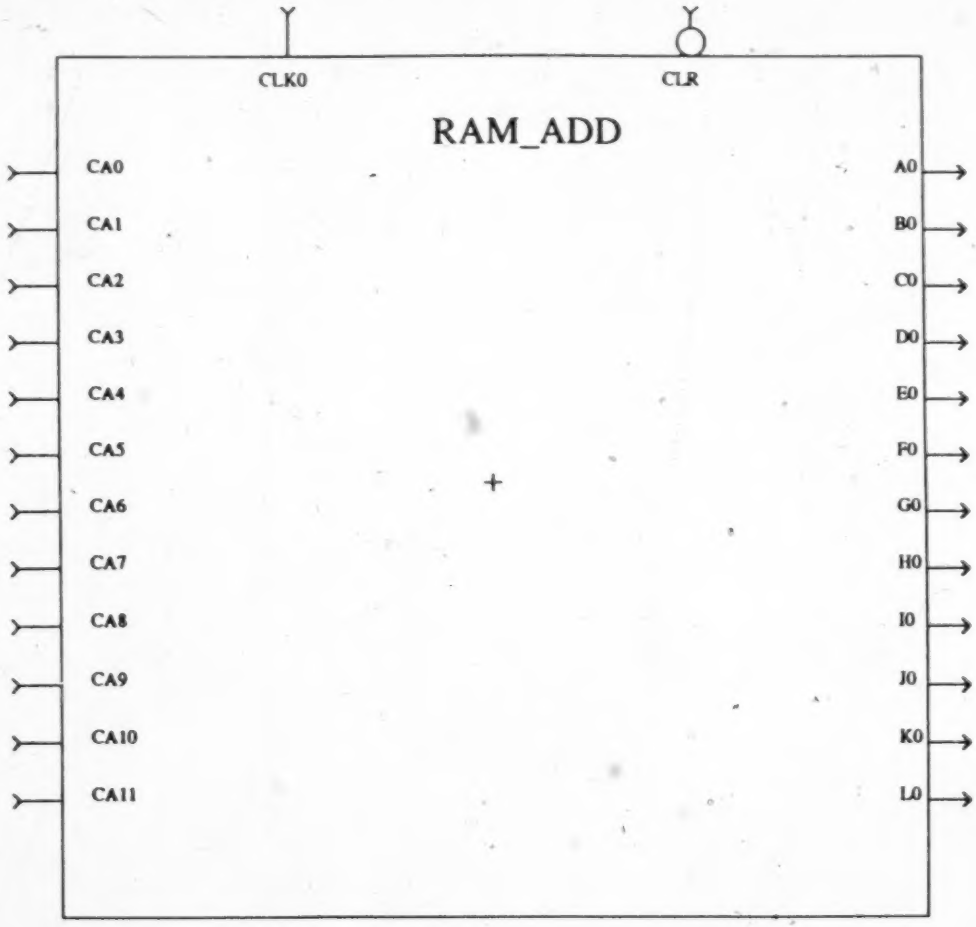
U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: SR4 DATE: MAY 9 1989 PAGE: 59 OF 63



U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: ADD_SEL DATE: MAY 9 1989 PAGE: 60 OF 63

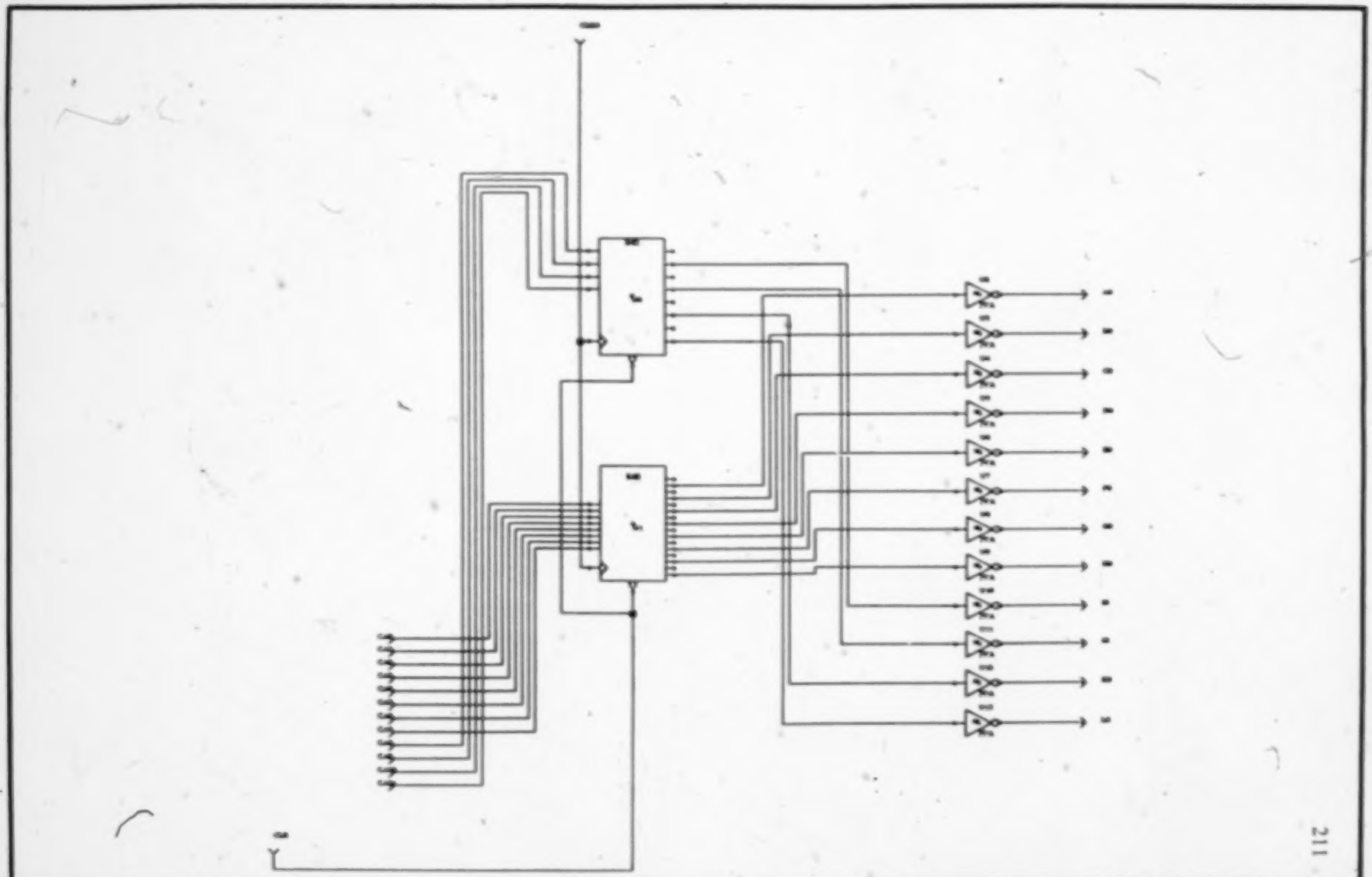


U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: ADD_SEL DATE: MAY 9 1989 PAGE: 61 OF 63



210

U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE ARRAY DESIGN	MODULE: RAM_ADD DATE: MAY 9 1989 PAGE: 62 OF 63



U OF A (E.E.)	DRAWN BY: M. PARANJAPE CHK'D:	TITLE: M.SC. PROJECT A DATA ACQUISITION GATE-ARRAY DESIGN	MODULE: RAM_ADD DATE: MAY 9 1989 PAGE: 63 OF 63

APPENDIX C

SIMULATION INPUT FILE

```
/*
 *
 *    SCL FOR NETWORK <FINAL1>
 *
 */
      DISPLAY
        INTCLK=U25/Z,,
        TRIGGER,,
        SIGNAL=HEX(DATAIN1.7,DATAIN1.6,DATAIN1.5,DATAIN1.4,DATAIN1.3,DATAIN1.2,DATAIN1.1,DATAIN1.0),,,,
        PRE_CNTR=HEX(U2/U1/U1/A.10,U2/U1/U1/A.9,U2/U1/U1/A.8,U2/U1/U1/A.7,U2/U1/U1/A.6,U2/U1/U1/A.5,U2/U1/U1/A.4,U2/U1/U1/A.3,U2/U1/U1/A.2
        POST_CNTR=HEX(U2/U1/U1/B.10,U2/U1/U1/B.9,U2/U1/U1/B.8,U2/U1/U1/B.7,U2/U1/U1/B.6,U2/U1/U1/B.5,U2/U1/U1/B.4,U2/U1/U1/B.3,U2/U1/U1/B.
        MEM_CNTR=HEX(U2/U1/U11/A.10,U2/U1/U11/A.9,U2/U1/U11/A.8,U2/U1/U11/A.7,U2/U1/U11/A.6,U2/U1/U11/A.5,U2/U1/U11/A.4,U2/U1/U11/A.3,U2/U
        DPR_ENAB=DIS_CE,,
        DPR_ADDR=HEX(U2/U1/U11/Z10,U2/U1/U11/Z9,U2/U1/U11/Z8,U2/U1/U11/Z7,U2/U1/U11/Z6,U2/U1/U11/Z5,U2/U1/U11/Z4,U2/U1/U11/Z3,U2/U1/U11/Z2
        DPR_DATA=HEX(DPR_DATA.7,DPR_DATA.6,DPR_DATA.5,DPR_DATA.4,DPR_DATA.3,DPR_DATA.2,DPR_DATA.1,DPR_DATA.0),
        DPR_WE=U1/U22/U30/Z,,,,,,
        RAM_ENAB=RAM_CE,,
        RAM_ADDR=HEX(U2/U1/U32/L0,U2/U1/U32/K0,U2/U1/U32/J0,U2/U1/U32/I0,U2/U1/U32/H0,U2/U1/U32/G0,U2/U1/U32/F0,U2/U1/U32/E0,U2/U1/U32/D0,
        DATA0=HEX(U2/U1/U0/RAM1.7,U2/U1/U0/RAM1.6,U2/U1/U0/RAM1.5,U2/U1/U0/RAM1.4,U2/U1/U0/RAM1.3,U2/U1/U0/RAM1.2,U2/U1/U0/RAM1.1,U2/U1/U0
        WE0=RAM_WE.0,,
        DATA1=HEX(U2/U1/U0/RAM2.7,U2/U1/U0/RAM2.6,U2/U1/U0/RAM2.5,U2/U1/U0/RAM2.4,U2/U1/U0/RAM2.3,U2/U1/U0/RAM2.2,U2/U1/U0/RAM2.1,U2/U1/U0
        WE1=RAM_WE.1,,
        DATA2=HEX(U2/U1/U0/RAM3.7,U2/U1/U0/RAM3.6,U2/U1/U0/RAM3.5,U2/U1/U0/RAM3.4,U2/U1/U0/RAM3.3,U2/U1/U0/RAM3.2,U2/U1/U0/RAM3.1,U2/U1/U0
        WE2=RAM_WE.2,,
        DATA3=HEX(U2/U1/U0/RAM4.7,U2/U1/U0/RAM4.6,U2/U1/U0/RAM4.5,U2/U1/U0/RAM4.4,U2/U1/U0/RAM4.3,U2/U1/U0/RAM4.2,U2/U1/U0/RAM4.1,U2/U1/U0
        WE3=RAM_WE.3,,,,,,
        EVEN=HEX(CH1EVEN.7,CH1EVEN.6,CH1EVEN.5,CH1EVEN.4,CH1EVEN.3,CH1EVEN.2,CH1EVEN.1,CH1EVEN.0),
        ODD=HEX(CH1ODD.7,CH1ODD.6,CH1ODD.5,CH1ODD.4,CH1ODD.3,CH1ODD.2,CH1ODD.1,CH1ODD.0),
        ARROW=HEX(ARROW.7,ARROW.6,ARROW.5,ARROW.4,ARROW.3,ARROW.2,ARROW.1,ARROW.0),,,,
        BLOCK=HEX(U1/U36/BLK3.2,U1/U36/BLK3.1,U1/U36/BLK3.0,U1/U36/BLSB),,,
        TIME_VAL=HEX(U1/U36/TIME4,U1/U36/TIME3,U1/U36/TIME2,U1/U36/TIME1,U1/U36/TIME0)
```

```
;

SAVE #LTEST ;

PRINT CHANGES ;

AT 16405000:
BEGIN
        TYPE "SIMULATION REACHED MAXIMUM SIM TIME: NORMAL EXIT";
        NOTE "SIMULATION REACHED MAXIMUM SIM TIME: NORMAL EXIT";
EXIT;
END

AT 0: NOTE "
******* THIS SIMULATION DEMONSTRATES THE CAPABILITY OF CAPTURING A SINGLE WAVEFORM IN THE ENTIRE 16K MEMORY BLOCK.
       STORAGE BEGINS IN BLOCK 0 USING THE ROLL-MODE AT A 10MHZ SAMPLING RATE.  THE TRIGGERING IS NOT
  CRITICAL, BUT THE FACT THAT CONTINUOUS CAPTURE IS BEING USED IS VITAL. *******
        ";

AT 0: NOTE "
  INITIALIZATION:
                RESET=0
                CLR_VAR=0
                XCLK=0
                EXTCLK=0
                TIMESEL=0
                UPDOWN=1
                BLKLOAD=0";
AT 0: NOTE "
  CONTROL LINE STATUS:
                ROLL=0          STORE=0         PRE25=0         LOCK_DPR=1      BLSB=0
                REFRESH=1       CLEAR=0         PRE50=0         REL_DPR=1       BLK3.0=0
                CONT_CAPT=0     RECALL=0        PRE100=0        HOLD=1          BLK3.1=0
                SINGLE_SHOT=1   LONG_REC=1      MEM_SEL=1       REL=1           BLK3.2=0
                RESET_MODE=1    STORELEV=0      CAP_DIS=0";
AT 0: NOTE "
  INPUT STATUS:
                TRIGGER=0
                EXECUTE=1";


AT 1000: TYPE "THE THING'S STARTED";

AT 0: SET DATAIN1.7=0;
AT 0: SET DATAIN1.6=0;
AT 0: SET DATAIN1.5=0;
AT 0: SET DATAIN1.4=0;
AT 0: SET DATAIN1.3=0;
AT 0: SET DATAIN1.2=0;
AT 0: SET DATAIN1.1=0;
```

```
AT 0: SET DATAIN1.0=1;

AT 0: SET RESET=0;

AT 0: SET CLR_VAR=0;

AT 0: SET XCLK=0;
AT 0: SET TIMESEL=0;
AT 0: SET EXTCLK=0;

AT 0: SET PRE25=0;
AT 0: SET PRE50=0;
AT 0: SET PRE100=0;

AT 0: SET UPDOWN=1;

AT 0: SET EXECUTE=1;
AT 0: SET TRIGGER=0;

AT 0: SET STORELEV=0;
AT 0: SET STORE=0;
AT 0: SET CLEAR=0;
AT 0: SET RECALL=0;

AT 0: SET ROLL=0;
AT 0: SET REFRESH=1;
AT 0: SET CONT_CAPT=0;
AT 0: SET SINGLE_SHOT=1;


AT 0: SET LOCK_DPR=1;
AT 0: SET REL_DPR=1;
AT 0: SET HOLD=1;
AT 0: SET REL=1;

AT 0: SET LONG_REC=1;

AT 0: SET RESET_MODE=1;


AT 0: SET MEM_SEL=1;
AT 0: SET CH1CAP_DIS=0;

AT 0: SET BLSB=0;
AT 0: SET BLK3.0=0;
AT 0: SET BLK3.1=0;
AT 0: SET BLK3.2=0;


AT 3000: NOTE "
         CLR_VAR=1";
```

```
      AT 3000: SET CLR_VAR=1;

      AT 4000: NOTE"
        CONTROL LINE STATUS:
                        STORE=1";

      AT 4000: SET STORE=1;

      AT 3500: NOTE "
                RESET=1
                BLKLOAD=0";

      AT 3500: SET BLKLOAD=0;
      AT 3500: SET RESET=1;

      AT 5000: NOTE"
        CONTROL LINE STATUS:
                        BLKLOAD=1
                        STORELEV=1";

      AT 5000: SET BLKLOAD=1;

      AT 5000: SET STORELEV=1;

      AT 5500: NOTE"
        CONTROL LINE STATUS:
                        STORE=0
                        RECALL=0";

      AT 5500: SET STORE=0;
      AT 5500: SET RECALL=0;

      AT 4500: SET TIMESEL=1;
      AT 5500: SET TIMESEL=0;


      AT 10000: NOTE "
                INPUT:
                        EXECUTE=0";

      AT 10000: SET EXECUTE=0;

      AT 11000: NOTE "
                INPUT:
                        EXECUTE=1";

      AT 11000: SET EXECUTE=1;

      AT 10000: NOTE " THE 20 MHz. EXTERNAL CLOCK IS ENABLED";

S1:   AT 10000: SEQ XCLK=0(250,500,E);
```

```
AT 100000: TYPE " 100000 ";

AT 300000: TYPE "TRIGGER ARRIVES HERE";
AT 300000: NOTE "TRIGGER ARRIVES HERE";

AT 300000: SET TRIGGER=1;
AT 350000: SET TRIGGER=0;



AT 16400000: TYPE "CIRCUIT IS RESET";
AT 16400000: SET RESET=0;
AT 16400000: CANCEL S1;

/*

AT 1500000: NOTE "
******* THIS SIMULATION DEMONSTRATES THE SINGLE-SHOT CAPTURE OF TWO WAVEFORMS IN A 1K AND 2K MEMORY BLOCK.
        FOR THE SECOND CASE, STORAGE IS IN BLOCK 0 USING THE ROLL-MODE AT A 10MHZ SAMPLING RATE.  NO
  PRETRIGGER INFORMATION IS REQUIRED IN THIS SITUATION. *******
        ";

AT 1500000: SET PRE25=0;
AT 1500000: SET PRE50=0;
AT 1500000: SET PRE100=0;

AT 1500000: SET UPDOWN=1;

AT 1500000: SET EXECUTE=1;
AT 1500000: SET TRIGGER=0;

AT 1500000: SET STORELEV=0;
AT 1500000: SET STORE=0;
AT 1500000: SET CLEAR=0;
AT 1500000: SET RECALL=0;

AT 1500000: SET ROLL=0;
AT 1500000: SET REFRESH=1;
AT 1500000: SET CONT_CAPT=1;
AT 1500000: SET SINGLE_SHOT=0;


AT 1500000: SET LOCK_DPR=1;
AT 1500000: SET REL_DPR=1;
AT 1500000: SET HOLD=1;
AT 1500000: SET REL=1;

AT 1500000: SET LONG_REC=0;

AT 1500000: SET RESET_MODE=1;
```

```
AT 1500000: SET MEM_SEL=1;
AT 1500000: SET CH1CAP_DIS=0;

AT 1500000: SET BLSB=0;
AT 1500000: SET BLK3.0=0;
AT 1500000: SET BLK3.1=0;
AT 1500000: SET BLK3.2=1;


AT 1503000: NOTE "
          CLR_VAR=1";

AT 1503000: SET CLR_VAR=1;

AT 1504000: NOTE"
  CONTROL LINE STATUS:
                  STORE=1";

AT 1504000: SET STORE=1;

AT 1504500: SET TIMESEL=1;
AT 1505500: SET TIMESEL=0;

AT 1503500: NOTE "
          RESET=1
          BLKLOAD=0";

AT 1503500: SET BLKLOAD=0;
AT 1503500: SET RESET=1;

AT 1505000: NOTE"
  CONTROL LINE STATUS:
                  BLKLOAD=1
                  STORELEV=1";

AT 1505000: SET BLKLOAD=1;

AT 1505000: SET STORELEV=1;

AT 1505500: NOTE"
  CONTROL LINE STATUS:
                  STORE=0
                  RECALL=0";

AT 1505500: SET STORE=0;
AT 1505500: SET RECALL=0;
AT 1510000: NOTE "
          INPUT:
                  EXECUTE=0";

AT 1510000: SET EXECUTE=0;
```

```
      READING IS FROM BLOCK 1 USING THE ROLL-MODE AT A 10MHZ SAMPLING RATE.  THE TRIGGER
  IS NOT INVOLVED IN THIS PROCESS. *******
        ";


AT 6410000: SET PRE25=1;
AT 6410000: SET PRE50=0;
AT 6410000: SET PRE100=0;

AT 6410000: SET UPDOWN=1;

AT 6410000: SET EXECUTE=1;
AT 6410000: SET TRIGGER=0;

AT 6410000: SET STORELEV=0;
AT 6410000: SET STORE=0;
AT 6410000: SET CLEAR=0;
AT 6410000: SET RECALL=1;

AT 6410000: SET ROLL=0;
AT 6410000: SET REFRESH=1;
AT 6410000: SET CONT_CAPT=1;
AT 6410000: SET SINGLE_SHOT=0;


AT 6410000: SET LOCK_DPR=1;
AT 6410000: SET REL_DPR=1;



AT 6414000: SET STORE=1;

AT 6414500: SET TIMESEL=1;
AT 6415500: SET TIMESEL=0;

AT 6413500: NOTE "
          RESET=1
          BLKLOAD=0";

AT 6413500: SET BLKLOAD=0;
AT 6413500: SET RESET=1;

AT 6415000: NOTE"
  CONTROL LINE STATUS:
                BLKLOAD=1
                STORELEV=1";

AT 6415000: SET BLKLOAD=1;

AT 6415000: SET STORELEV=1;

AT 6415500: NOTE"
  CONTROL LINE STATUS:
                STORE=0
                RECALL=0";

AT 6415500: SET STORE=0;
```

```
      AT 6415500: SET RECALL=0;

      AT 6416000: NOTE"
        CONTROL LINE STATUS:
                        RECALL=1";

      AT 6416000: SET RECALL=1;


      AT 6420000: NOTE "
                INPUT:
                        EXECUTE=0";

      AT 6420000: SET EXECUTE=0;

      AT 6421000: NOTE "
                INPUT:
                        EXECUTE=1";

      AT 6421000: SET EXECUTE=1;

      AT 6420000: NOTE " THE 20 MHz. CLOCK IS ENABLED";

S4:   AT 6420000: SEQ XCLK=0(250,500,E);


      AT 12800000: TYPE "CIRCUIT IS RESET";
      AT 12800000: SET RESET=0;
      AT 12805000: CANCEL S4;


 */

S2:   WHENEVER (U25/Z=1)

                CALL PATS;
                SIMULATE FOR 1000;

      GOTO S2;


PATS: PATTERN PAT DATAIN1.7,DATAIN1.6,DATAIN1.5,DATAIN1.4,DATAIN1.3,DATAIN1.2,DATAIN1.1,DATAIN1.0;
      CANCEL S2;
      RETURN;


      SIMULATE UNTIL 16405000;
```

```
DONE: EXIT;
```